AN ABSTRACT OF THE THESIS OF

John H. Poland for the degree of Master of Science in Mechanical Engineering presented on August 17, 1993

Title : A Computer Imaging Method for Fluid Motion Studies In Metal Casting

Abstract Approved: Redacted for Privacy

Lorin R. Davis

Significant reductions in the number of defects in parts produced by investment casting can be obtained by improving the flow of the molten metal during pouring. Studies have been done at Oregon State University with simulated casting techniques to determine optimal mold configurations. Better mold layouts have been shown to improve this flow.

The purpose of this project was to develop a computer imaging system that would aid in reliably evaluating these experiments. The completed system consisted of a computer, a frame grabber, video equipment and operating software. Videotapes made of the fluid flow in the mold during the pouring process were replayed into the computer and evaluated. Custom software reduced the collected data to a representative evaluation number.

Results show that the computer evaluations are reliable and reproducible, but applications are limited because of the cost, speed and power of available computer systems.

A Computer Imaging Method for Fluid Motion Studies in Metal Casting

by

John H. Poland

A Thesis

submitted to

Oregon State University

in partial fulfillment

of the requirements

for the degree of

Master of Science

Completed August 17, 1993

Commencement June 1994

APPROVED:

Redacted for Privacy

Professor of Mechanical Engineering in charge of major

Redacted for Privacy

Head of Department of Mechanical Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented August 17,1993

Typed by John H. Poland for John H. Poland

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

A Computer Imaging Method for Fluid Motion Studies in Metal Casting

INTRODUCTION AND BACKGROUND

The Mechanical Engineering department at OSU researches methods to improve investment casting through several projects funded by the Oregon Metals Initiative (OMI). Mark Miller completed one major project in 1991. Mr. Miller studied the effects that mold configurations have on turbulence and fill time in the investment casting process.

Investment casting is done in several steps. A ceramic mold is created by the lost-wax method. The mold consists of a tundish, downsprue, horizontal and vertical runners, and gates as shown in figure 1. The mold is preheated and the molten metal poured into it. Once the metal cools, the ceramic mold is broken away, and the cast parts are cut from the risers and gates [1]*.

Excessive turbulence and poor flow control create problems during the pouring process. The molten metal can erode the mold walls, leading to dimensional changes in the part and causing impurities from the eroded mold walls to enter the metal. Bubbles, splashing and flow separation

---

* Numbers in [] refer to references

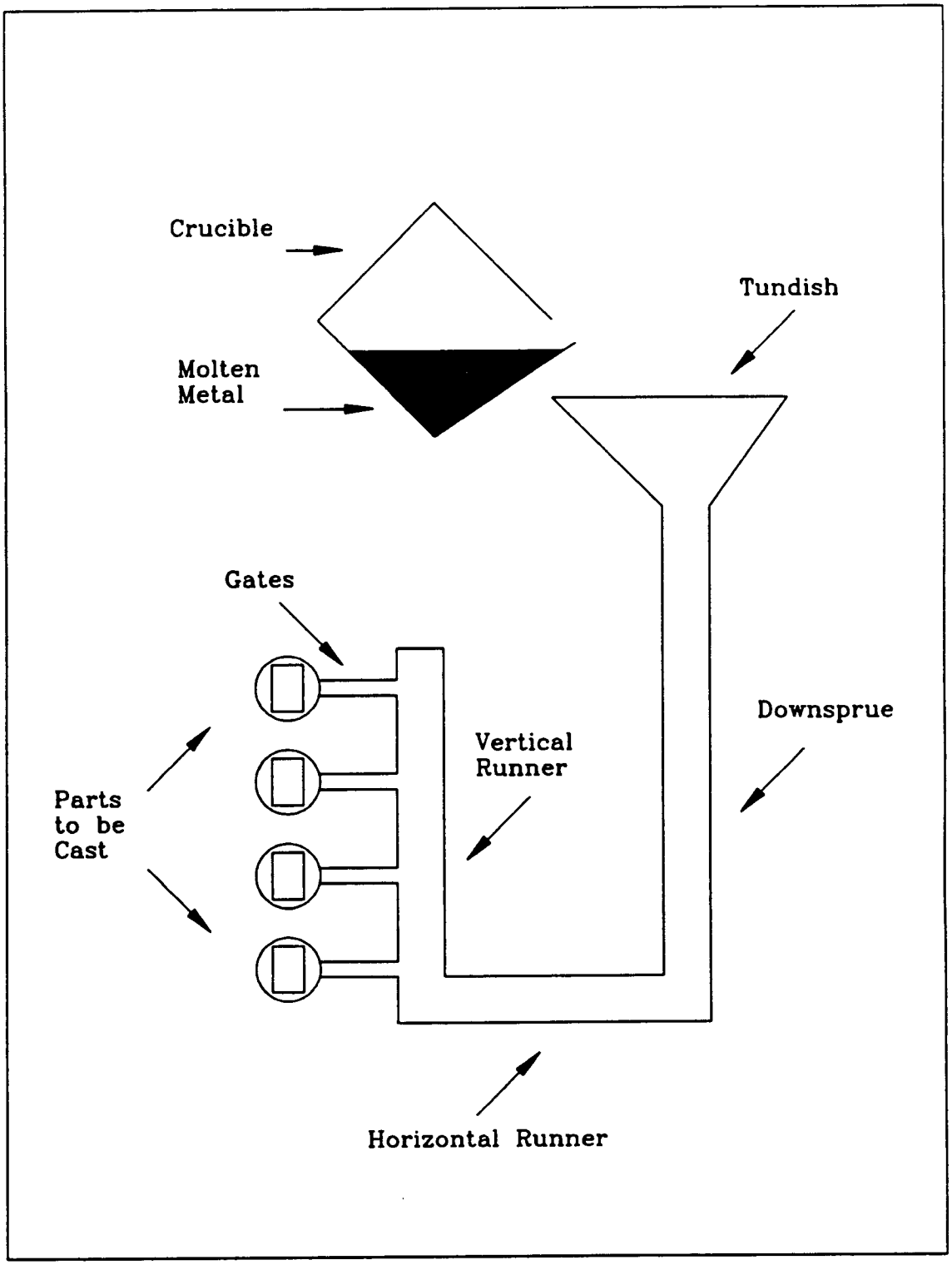

Figure 1. Typical Investment Casting Mold

in the flowing metal can lead to discontinuities, defects, and porosity. Because of these problems the part must be reworked to produce a complete, finished piece [2]. The goal of Miller's investigation was to improve the flow of metal in the mold during the pouring stage, and thus reduce the defects. Several important conclusions were reached regarding the use of chokes and wells, and the effect of pour rate. Specific recommendations were made to improve casting quality based on these conclusions.

The experiments were done by modeling the basic mold configuration with plastic pipe sections and cast plastic runner sections. A simplified setup was used to give general, consistent results. Water was used to model the molten metal. It has a viscosity similar to that of molten titanium, but has a density and surface tension much lower than molten metals. The relevant dimensionless scaling parameters are :

Froude number (for open channel flow) = v^2/gl

Reynolds number (for closed conduit flow) = $lv/\mathbf{n}$

Weber number (for surface tension effects) = $\mathbf{r}v^2l/\mathbf{t}$

where :

g = Acceleration of gravity
l = Characteristic length
r = Fluid density
v = Fluid velocity
n = Fluid viscosity
t = Surface tension coefficient

These parameters can be thought of as ratios of forces involved in the fluid flow. The Froude number is the ratio of inertia force to gravity force, the Reynolds number is the ratio of inertia force to viscous force, and the Weber number is the ratio of inertia force to surface tension.

The experimental molds were constructed full size, so good agreement was achieved for the Reynolds number and Froude number. The Weber number differs greatly due to the lower surface tension of water. Some typical values of fluid properties and the scaling parameters are :

Water at 293K [3]:
density = 1.00 gram/cc (1000 kg/m3)
visosity = 1.00 mPa-s (**n**=1x10E-6 m2/s)
surface tension = 0.073 N/m

Titanium [4]:
density = 4.1 gram/cc (4100 kg/m3)
viscosity = 0.62 cp (**n**=0.6x10E-6 m2/s) at 2033K
surface tension = 1588 dynes/cm (1.6 N/m) at 1900K

For a typical pouring condition of 1" diameter pipe and 1 ft/sec flow velocity, the scaling parameters are :

Water :
Froude number = 0.37
Reynolds number = 7,700
Weber number = 32

Titanium :
Froude number = 0.37
Reynolds number = 13,000
Weber number = 6.0

It was decided that water would be an acceptable model since the greater surface tension of the metal would

increase the quality of the pours. What worked well for the water test case should work better for the actual metal. This was shown to be true for a comparison case done with molten solder [5].

Mold configurations were tested for effect on the fluid flow. The downsprue and horizontal runners were modified by adding wells, chokes and tapered sections as shown in figure 2, page 7. The experiments were videotaped for later analysis and rated according to an evaluation scheme developed by the researchers to compare the various configurations. A rating number was assigned to each configuration based on turbulence, bubbles in the fluid, and smoothness of the flow. This rating scheme is given in Table I, page 8. Evaluations of the experiments were compiled into recommendations as to the best configurations, and the results presented to Precision Castparts Corp. of Portland, Oregon in 1991. The general conclusions, taken from the work of Mark Miller [6], were :

* Attention is first drawn to the importance of the pour rate. It was noticed that most of the best pours were made at the slowest pour rate and that the majority of the poorer runs were made at the fast pour rate. The medium pour rate data fell in the middle. This tendency of slower pours to be smoother was expected because of the smaller turbulence induced in the flow.

* The shape of the downsprue, whether tapered or straight, did not seem to be important as long as the minimum area was small enough to provide the required [flow] choking.

* The use of a well contributed to the smoothness of the flow, but its effect can be provided instead by chokes.

* Overflow chokes in the horizontal portions of the mold seemed to be more effective than chokes in the downsprue, but the use of both improved the flow.

* While runner extensions and underflow chokes might help to reduce impurities, they did little to help the smoothness of the flow.

These observations and the rating scheme used to obtain them were both considered as foundational during development of the present project.

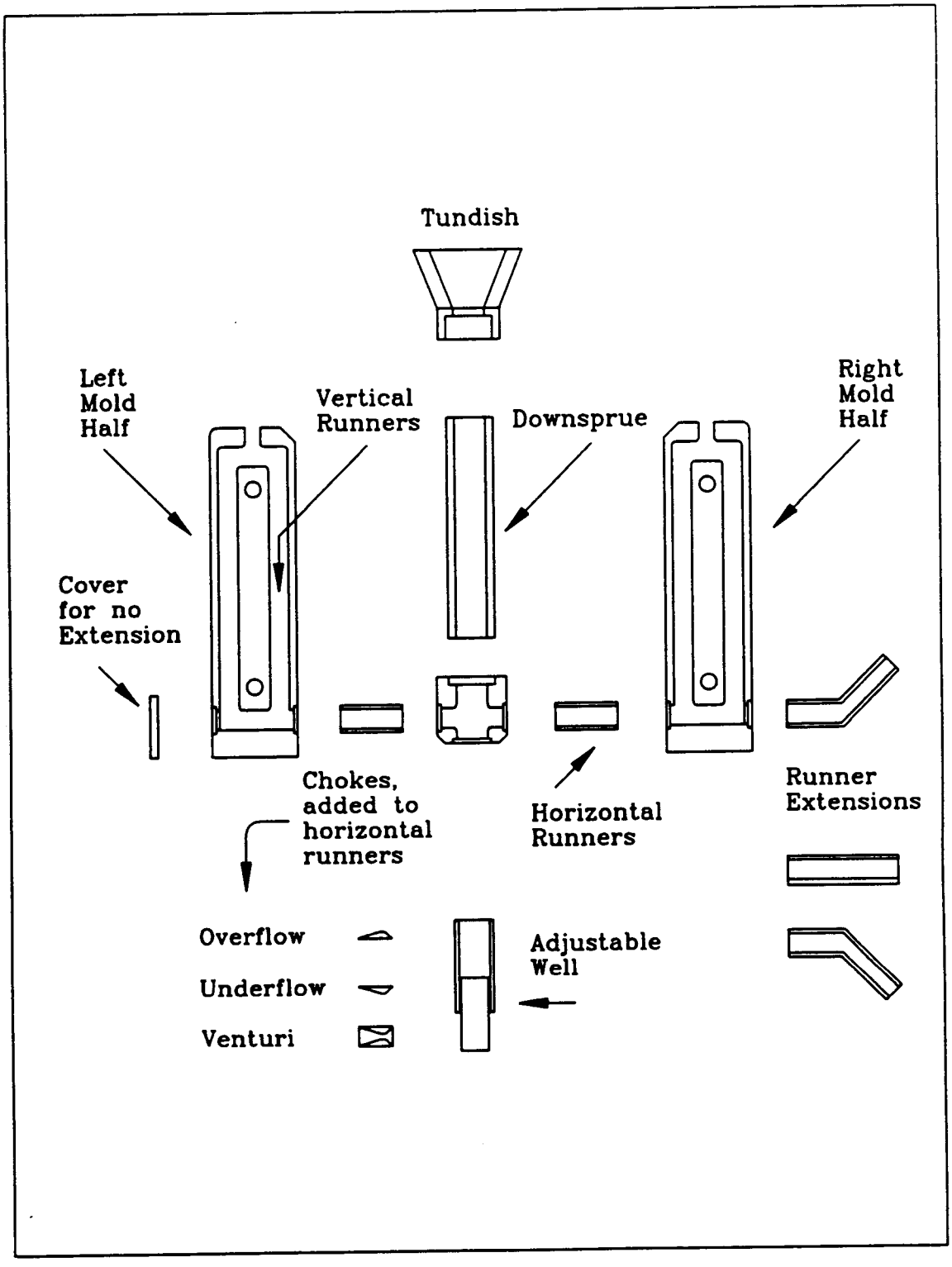

Figure 2. Model of Mold Used in Project

TABLE I. Flow Evaluation Rating Scheme

FLOW EVALUATION WORKSHEET v3.0 (1991)

NOMENCLATURE

MINOR 1 point each	MODERATE 10 pts ea
LT - Light turbulence	FLB - Few Large Bubbles
FVB - Few Visible Bubbles	LVB - Lots Vis Bubbles
SW - Swirl	USF - UnSteady Flow
	UF - UnEven Flow (UF)

SERIOUS 100 points each	Post Fill 10 pts ea
HT - Heavy Turbulence	HT
FS - Flow Separation	
LLB - Lots of Large Bubbles	LLB
SP - Splash	SP
SL - Slosh	SL
JT - Jetting	

DEFINITIONS

Light Turbulence - perceptible tumbling motion of fluid
Few Visible Bubbles - five or fewer bubbles, less than 1/4 of the width of the runner
Swirl - cycloidal motion of fluid

Unsteady Flow - noticeable acceleration in runner fluid level
Few Large Bubbles - one bubble at least as large as the width of runner
Uneven Flow - Fluid column in both runners on one side not filling at equal height

Heavy Turbulence - SERIOUS - includes flow reversals (FR), the change in direction of fluid flow in a runner

- POST FILL - includes flow reversals, double sloshes (DS), or flow separations

PURPOSE AND OBJECTIVES

One goal of the continuing metal casting research at OSU was to develop an automated method of studying the fluid flow experiments. Specifically, a computer imaging system was needed for more simple and consistent evaluations of the metal casting experiments. The visual evaluations were good, but depended on several subjective factors. Different evaluators would notice different flow problems. The same evaluator would notice different things at different times. If the project was allowed to lapse and then restarted with new people, there would be little continuity in the rating of subjective problems, such as how big of a splash from a bursting bubble was significant. Visual evaluations are also time-consuming and tedious. The potential consistency and reliability of visual imaging and computer analysis makes an automated method quite useful.

Work began on such a method in the summer of 1991. The goals for this computer imaging project were:

1. Develop a reliable, easy-to-use system to evaluate the fluid flow experiments.

2. Obtain evaluation numbers for a representative sample of the experiments done.

3. Compare the results obtained with the computer system to those done visually.

Once specific goals were established, research began into the computers and imaging systems available. Because of the costs and development time involved a simple, relatively inexpensive system was needed. Several options were available at the time of the start of the project. A hardware-oriented system from Imaging Technology, Inc. of Bedford, Massachusetts had suitable computing power, and could be driven directly by user-developed computer code. Such a system is fast, but would have required extensive software development, and the cost was beyond the budget recommendations. Other systems are based on prepackaged software. They can be adapted quickly to image processing, but didn't have a large selection of built-in features suitable for an automated system. They also were beyond budget. A compromise was reached with a software-based system that was oriented toward the development of automated analysis routines. The system was DOS-based with a lower-cost frame grabber and operating software from Data Translation of Marlboro, Massachusetts. Since this setup was chosen, the industry has produced much more powerful and varied systems. The developer of a new system would want to review all of the options before making a decision.

One system tested was Macintosh-based with a QuickCapture frame grabber from Data Translation. This system was easy to use and allowed still-frame image capture and analysis from a VCR. It could be used with Image 1.44, a shareware program obtained from the area computer network. Its chief limitation when used in this form was the great difficulty in collecting the needed data and sending it to a file rapidly enough. It is a good system for extensive analysis of a few images.

In addition to the packaged software for operating the video system, several data-analysis programs had to be written for the project. The data-gathering procedure generates a great volume of raw image data that must be reduced to give a usable evaluation number for the experiment.

A high-speed camera capable of filming at 500 pictures per second was tested by the Mark Miller group. Three special runs were made. These runs were evaluated by the visual and computer methods to determine its usefulness for this project. Only a few experiments were recorded this way as part of the original project. Most data were recorded with the standard camcorder. This project was oriented toward reevaluating these original experiments with the computer method, but tests using the high-speed camera were made to evaluate it for future use. The high speed camera was shown to simplify the analysis procedure for the computer method, but has the limitation of greatly

reduced image contrast. Comparisons of the images obtained from an experiment by both cameras show the high-speed camera images to be of much less contrast. This reduced the effectiveness of the computer evaluations. Data from the comparisons is shown in the results section.

SYSTEM CONFIGURATION

Hardware

1. Computer

   An IBM-compatible 80486DX-based computer running at 33MHz.

2. Frame Grabber

   A DT 2867-LC frame grabber from Data Translation of Marlboro Massachusetts.

3. Panasonic VHS VCR model AG-2510
4. Panasonic VHS Camcorder model AG-170
5. Sony Trinitron Monitor model PVM-2530
6. Miscellaneous cabling and connectors

Software

1. Microsoft Windows v3.0
2. Global Lab Image v1.01 from Data Translation
3. Software written in Pascal for the project to reduce collected computer data to evaluation numbers.

EQUIPMENT LAYOUT

The schematic layout of the equipment is shown in figure 3. Data collection was done in several steps. The experiment was first filmed with the camcorder. A slow-motion copy of the videotape was played back on the VCR, connected to the computer through the frame grabber. As the tape played, the analog signal from the VCR was digitized and displayed by the frame grabber. The video system software allowed a particular picture to be 'captured', or stored in video memory for further study. The complete system as used included the 'script' program, which collected data from each captured frame and sent it to a disk file for further reduction. Evaluations were done by playing back the tape of the experiment while running the script file to collect data.

The signals from the VCR and camcorder conform to the RS-170 video standard for television signals [7]. This standard calls for the television picture to be displayed as a series of frames, one every 1/30 second. To produce the picture an electron beam is scanned on the television picture tube. Each frame is made up of two interlaced or overlapping 'fields'. The two fields are slightly offset in position from each other and are displayed sequentially. Because a new field is displayed every 1/60 second, motion in a television picture appears smooth to

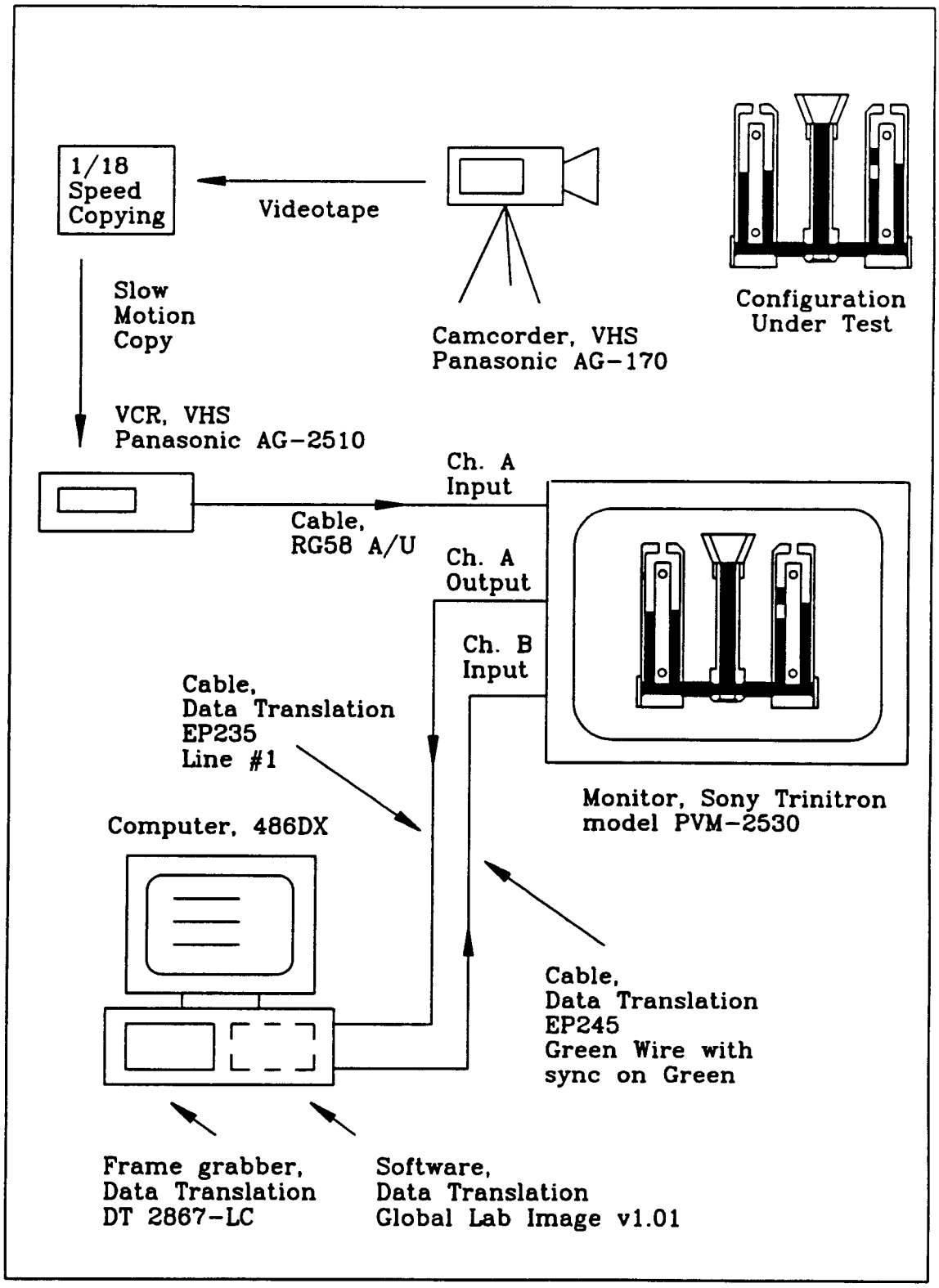

Figure 3. Equipment Layout Schematic

the eye, which cannot sense changes in the image faster than about 1/16 second [8].

The scan of the electron beam determines the position of a displayed point, and the intensity of the signal (voltage level) determines the brightness. Color information is added also, but the frame grabber used for this project is monochrome only, therefore shades of gray in the black-and-white image provided all of the image data. Color information can help a video analysis by adding additional data, but obtaining it requires more complex and expensive equipment.

Once digitized, the locations and brightness (gray level) are stored in an array in video memory. The DT-2867 frame grabbers allow the television signal to be digitized continuously into an array of points 640 wide X 480 high, eight bits per point. Each point represents a pixel, the smallest element of a digital picture. Eight bits provides 256 possible gray levels for each pixel, the value '0' representing black, and '255' representing white. This digital information can be stored as a file and later displayed or analyzed as data by computer programs such as the script files mentioned previously [9].

The level of brightness as indicated by voltage level from the camera can be adjusted in software. This allows the frame grabber to be adjusted to compensate for very bright or very dark images. The standard settings were used throughout this project.

EQUIPMENT IMPLEMENTATION

Global Lab Image (GLI) has provisions for creating script files in interpreted 'C'. These script files are macros that will automatically run image processing routines. They were created by stepping through the analysis routine with the 'script recorder' function of GLI. The general form of the script file was generated automatically by the software. The file was then modified to include specific routines needed for efficient data collection. Script files developed for this project captured an image and digitized it, laid out a series of profile lines, and sent this information to a disk file. The script looped through the process for a programmed number of iterations while the videotape ran. Data was collected from the videotape while the tape played.

The 'Profile Line' function of GLI collects the value of each pixel at each point along a line on the digitized image. The lines were programmed with beginning and ending coordinates, each point along the line corresponding to a pixel. Eight profile lines were used, giving pixel data for each major region of the image. Typical positioning of these lines is shown in figure 4. This pixel data was analyzed to determine where the fluid was in each region and to give a measure of its turbulence and bubble content. The same eight profile lines were used for each run. The addition of runner extensions and wells changes

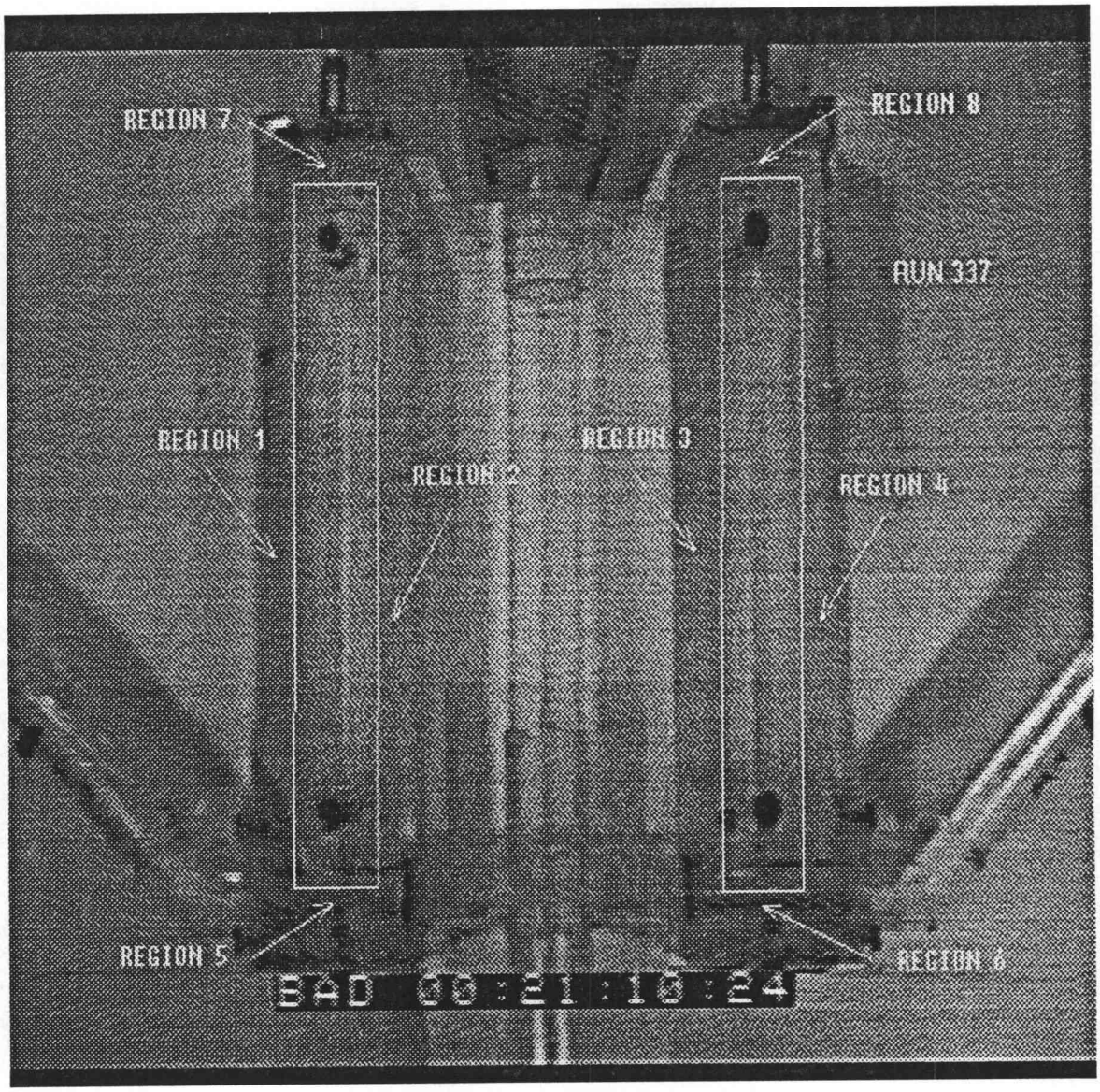

Figure 4. Regions and Profile Lines for Run 337

the flow pattern but these areas were not areas of evaluation.

The location of the lines was chosen separately for each individual run by trial and error fitting. The lines were placed as close to the center of each row and column as possible. A positive slope was chosen for the columns so that the horizontal coordinate (x) of the upper point would be greater than the lower point. Pixel data is read by the script in order of increasing 'x' coordinate, or from first point to last point if the 'x' coordinates are equal. This placement insured that pixel data is read in order from the bottom of the column to the top.

An example of the data collected by a profile line function is shown in figure 6. This graph represents the gray level of the pixels along the line for region 3 of the image in figure 5, from the bottom of the column to the top. The low values represent the dark fluid, the peak in the graph comes from the large bubble in the fluid, and the high values come from the light-colored background. Analysis of these pixels provided the data for the height of the fluid in the column (the 'edge') and the bubble/turbulence level. The more bubbles present, the higher the average pixel value in the fluid and the higher the standard deviation of the pixel values.

Approximately 1.8 seconds are required for the script file to digitize the image and collect and store the data. Because some fast runs were completed in less than 1.0

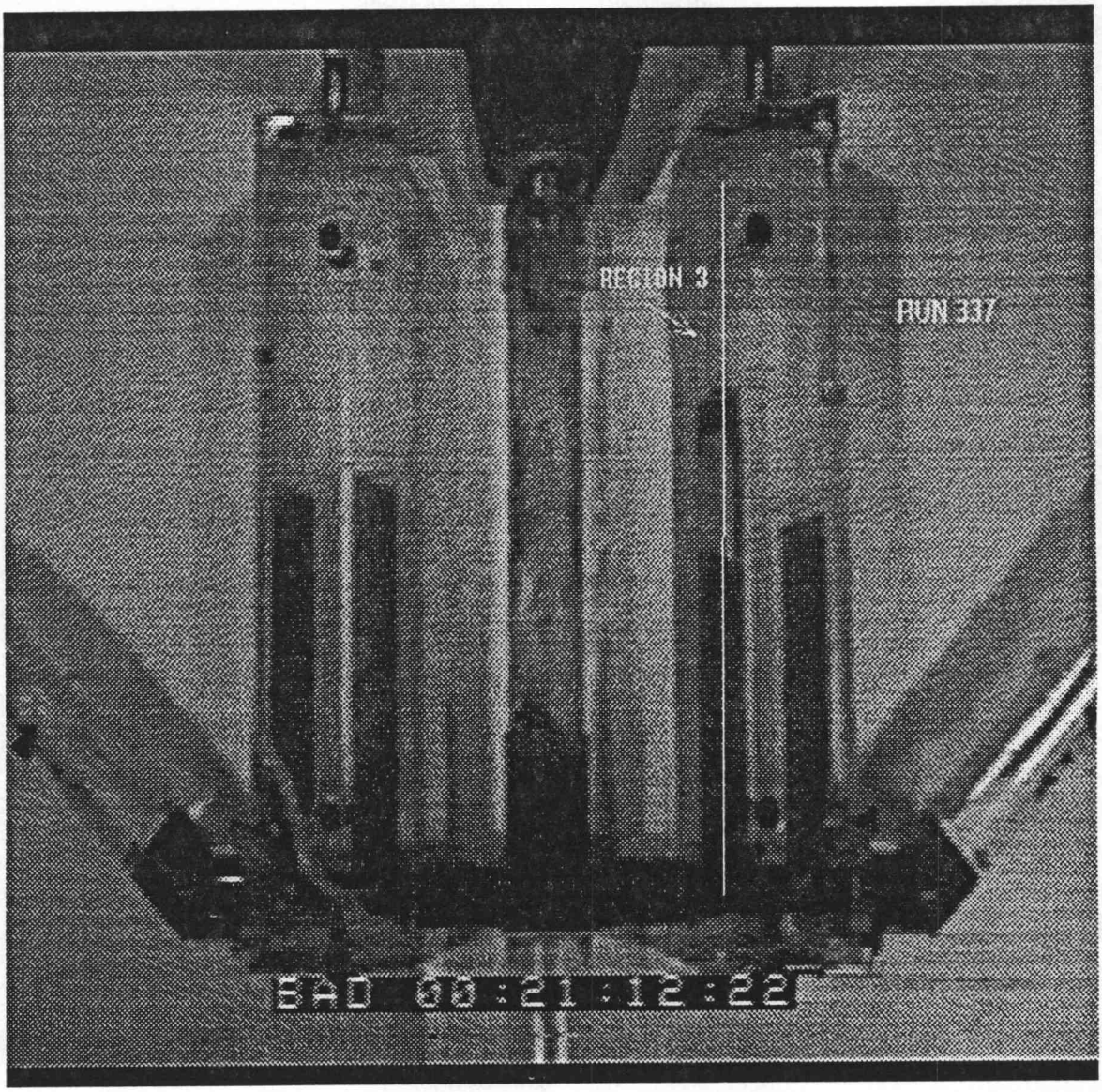

Figure 5. Partially Filled Mold

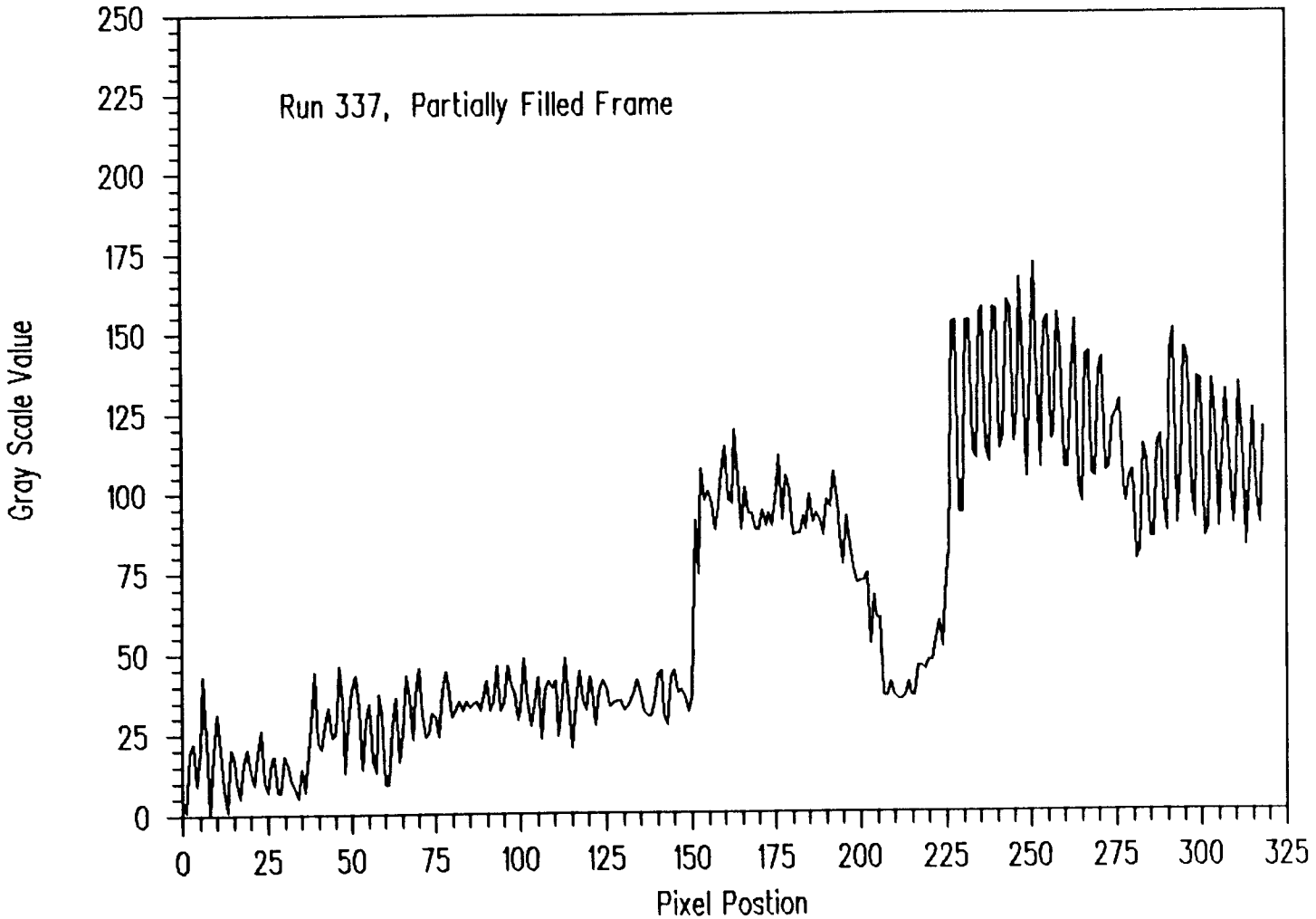

Figure 6. Graph of Profile Data from Run 337

second, much effort was put into developing a system to gather the data more rapidly. Methods such as pausing the VCR to collect still-frame data or running the VCR in slow-motion mode failed because of sync-signal instabilities. Several high-end VCR models were tried, but only when a time-base corrector was used was the sync signal stable enough for the computer to get a good image. This would have involved several thousand dollars more in equipment investment.

The problem was handled by copying the videotaped experiments onto another tape at slow speed. The Oregon State University Communications Media Center has one-inch tape equipment that allows copying at speeds ranging from 2X speed down to 1/18 speed. These machines are not calibrated, so approximate slow-motion speeds were obtained by using a stopwatch to time the images having recorded time bars and by comparison of slow-motion time with known actual times for several runs. The tapes were copied onto the one-inch format, played back at 1/18 speed, and copied onto another VHS-format tape. This was a very labor-intensive operation. It required about 5 hrs to copy 100 runs. A total of 556 runs was made originally. The original evaluations were done by copying the tapes to about 1/9 speed. This was adequate for visual evaluations. The tapes of the first 100 runs were of poor image quality, so they were not used for the computer evaluations. Runs 100-200 have no original copies and the 1/9 speed copies

have a black time bar (elapsed time indicator) superimposed across the middle of the images. Runs 147-156 were taken as a sample.

The data in the time bar region was dropped from the analysis, which shortened the apparent column height. The Pascal program 'RENUM' (Appendix E) was written to remove the time bar values from the run###.dat data files, where '###' is the run number. Less than 10% of the total data for the run was lost. Various other runs from the complete set were chosen as representative, some very good, some very bad, most others average.

Data files were made for each run. Script files were labeled as 'test###.scr' and generated pixel data files labeled 'run###.dat'. Once the script was loaded, the tape was started and the script activated when fluid entered the mold. Data collection was then completely automatic. The columns were about 350 pixels high and the rows about 80 pixels wide. Eight regions for each frame ,with up to 35 frames per run resulted in the data files requiring an average of 200 k-bytes of storage space each. Thirty-five M-bytes of hard drive space was required to store all the script files and the run data they generated. Floppy disk files could be used for backup, but not for initial storage. The script running time is doubled when floppy disks are used so only half the information would have been obtained.

For the fast runs, less than 10 frames were used in the analysis, which limited the accuracy of the analysis. Copying the tapes again to an even slower speed would solve this problem but only at the expense of greatly increased copying and data-gathering time.

Appendix A shows an example script file, for Run 337. Appendix B shows the data collected for column 3 (region 3) of the partially filled mold in figure 5, Run 337. Following is a step-by-step procedure for carrying out the analysis.

PROCEDURE FOR USE

1. Film the mold fill experiment with a suitable camera, such as a VHS or 8mm camcorder.

2. Copy the videotape to another tape at a much slower speed, such as the 1/18 speed available through the one-inch equipment at the CMC.

3. Connect the video equipment assembly as shown in figure 4. Use camera line #1 of the EP235 cable. The script files use video input #2, which corresponds to line #1 in hardware.

4. Enter Global Lab Image, under Microsoft Windows v3.0

5. Capture an image of the mold configuration under evaluation. Using the Profile Line function, determine the coordinates of the eight points required to place profile lines in the centers of the four rows and four columns, similar to figure 2.

6. Call up the script editor and a copy of an older script. Modify the script point constants, number of frames to check and data file name, and save the script itself under an appropriate name. Select all of the script with the editor and 'Evaluate' the script with the editor function to check for errors.

7. Exit the editor and Global Lab Image. Erase the file with the data file name, created by running the evaluation function. Presence of this file will prevent proper operation of the script. The file will be created again when the script is actually run, but it will then have the proper data.

8. Re-enter Windows and GLI. Load the script created in step 6. While playing the slow-motion videotape copy of the experiment, initiate the script by a mouse click on the file name when the fluid begins to enter the mold. The script will take data and store it to the hard drive automatically.

9. Check the data file created with a text editor capable of handling very large files, such as WordPerfect 5.2. If long sequences of zeros are noted, especially near the beginning of the file, the first frame is a bad data set and must be removed. Remove this first frame by counting eight "-1's." These are the end-of-region markers. Bad data sets occur frequently, due to overlapping of the image

capture and screen blanking intervals of the video system. It is a random occurrence and cannot be easily prevented.

10. Run the CALC.COM program on the edited program. Save the collected data summary, rinfo.### is default.

11. Run the EVAL.COM program on the data summary. The output of EVAL.COM shows the details of the evaluation, including fill time. If several files are to be evaluated, the EM.COM program will save the results and loop through input RINFO.DAT files.

12. Printouts of the digitized images may be obtained by saving them as generic TIFF files in Global Lab Image, without any compression schemes. These TIFF files can be printed out using several existing software packages that import TIFF graphics. For this project they were converted to Macintosh format and printed out using the Aldous PageMaker v4.0 desktop publishing program.

DATA REDUCTION

The CALC.PAS program was written to calculate some important values representing the fluid in each region. The 'edge' of the fluid flow (example - the height of the fluid in a vertical column), and the average and standard deviation of the pixel values in the fluid (those inside the edge) were determined for each frame. Determining the edge required comparison of the pixel values in the region with those representing the 'background', or the apparatus with no fluid in it. The first frame read into the CALC program must always be a background frame. An array of cutoff values was generated for each region from this data. The standard deviation of the pixels in the whole region was calculated. A subset of 9 pixels (those up through the pixel of interest) was averaged. The standard deviation, divided by the square root of the subset size (nine), was calculated for a 'cutoff difference'. This difference was then subtracted from the average of the subset value, to produce the cutoff value for that particular location.

Typically, a column would have a series of pixel values with the standard deviation equal to about 15. In a bright region of the background the average for a local group of nine pixels might be 180. The cutoff difference would then be 5, and the cutoff value for that location would then be 175. From statistics theory the probability of the average of the subset being greater than 175 in that sub-sample is

0.85 [10]. There was therefore an 85% chance that the edges were located within one pixel.

Several methods of edge location were tried. One method involved checking the slope of the gray scale values from the profile line. When a high slope was present, an edge was found. This method was rejected because of the similarity of bubble edges and fluid edges (see figure 5). Image processing operations that extract edge data from 2-D pictures such as the Laplacian filter could be effective. These types of methods use changes in the values of pixels and pixel line slopes to detect edges [11]. Image subtraction could also be effective. In this method the background common to all images is subtracted out. This leaves only the changes due to motion of the object in the image [12]. These methods would work but processing time is much too long. Collecting data from the profile line function was the only method available that was quick enough. Since the data collected was one-dimensional, only the slope and value change was left. False edges could result from illumination changes or bright spots in the image. Too high a cutoff value gave more false edges, while too low a value interpreted bubbles to be edges. The number of pixels in the subset and the division of the standard deviation by the square root of this value was chosen to be the best compromise. Extensive experimentation resulted in the present method, which has proven to be very reliable.

Once the cutoff values are determined, each succeeding frame is analyzed for edge information based on the cutoff. The pixel values were checked in order (from bottom to top in a column). At each location the average of the subset of the previous eight values and the current value was taken (total of nine values). If the average was greater than or equal to the cutoff value (175 in the example above) then the edge was located. The pixels up to the edge were then averaged and the standard deviation calculated. This information, for each region of each frame, along with the total length of each region, was then written to another disk file. In this program the run information files created by CALC are saved as 'rinfo###.dat'. This data was collected once for all four columns, and twice for all four rows. Splashing and sloshing in the horizontal runners as well as the top rows filling from both sides required that data be collected right-to-left and left-to-right for each row. The regions were numbered, 1-4 being the vertical columns (bottom-to-top only), 5-8 being rows checked left-to-right, and 9-12 being rows checked right-to-left. Appendix C contains a listing of CALC.PAS.

The 'rinfo' data was reduced to a single evaluation number with the EVAL.PAS program. This program compared the edge information for each vertical column to determine evenness of the fill and flow reversals. Each flow reversal was counted as 100 points, and each time an

unevenness in the fill was detected it was counted as 10 points. When a column was filled, the row above it was then checked in the appropriate direction. All regions considered logically appropriate (considering fluid levels) were checked for standard deviation. The standard deviation for each region was added to determine a total, then multiplied by a weighting factor. This factor was determined by running the EVAL.PAS program for run 503, which had no uneven fills or flow reversals, and calibrating it for an evaluation of 750. Run 503 is considered a good run. Using it as the basis made possible direct comparisons with the visual evaluations. Appendix D contains a listing of EVAL.PAS.

Fill times for the runs were calculated also. Each slow-motion frame represented 0.1 second, so the fill time could be determined from the number of frames counted from fluid entering the mold to fluid filling the mold. 'Filled' meant both top rows having both fluid edges at the opposite mold edges. The row then appeared filled with fluid, along that profile line.

Once the mold was filled, succeeding frames were analyzed for 'post-fill' problems. The procedure was the same as for 'fill', but the resulting evaluation score was divided by 10 before being added to the 'fill' results. Post fill problems were deemed much less serious. The total evaluation number and fill time was computed for each run. These results are shown in Table II.

Table II. Computer Evaluation Results

COMPUTER EVALUATIONS OF MOLD FILLING EXPERIMENTS
STANDARD DEVIATION UNWEIGHTED, RUN 503 USED AS BASIS
EVAL.COM, WITH WEIGHTINGS : W1=10, W2=100, W3=8.762

RUN #	FILL UF	FILL FR	POST FILL UF	POST FILL FR	COMPUTER TIME	COMPUTER EVAL	VISUAL TIME	VISUAL EVAL	EVAL % DIF
147	0	0	0	0	2.0	839	2.6	890	- 5.7
148	1	0	0	0	2.3	1050	3.0	820	28.0
149	0	0	0	0	2.2	931	3.1	790	17.8
150	3	0	0	0	1.6	1013	2.4	1020	- 0.7
151	1	0	0	0	1.6	1064	2.4	1130	- 5.8
152	0	0	0	0	1.8	1347	2.8	920	46.4
153	2	0	0	0	2.2	1150	3.2	800	43.8
154	1	0	0	0	2.3	1305	3.2	830	57.2
155	1	0	0	0	1.7	1317	2.5	990	33.0
156	2	0	0	0	1.5	1271	2.7	1110	14.5
280	2	0	0	0	2.4	1226	3.3	1340	- 8.5
281	4	2	0	0	2.2	1394	2.5	1360	2.5
282	6	0	0	0	1.5	1229	2.0	1460	-15.8
283	3	0	0	0	2.6	1173	3.6	870	34.8
284	3	0	0	0	3.0	1072	3.0	970	10.5
285	2	0	0	0	1.9	1038	2.6	1060	- 2.1
286	1	0	0	0	3.0	1106	4.1	940	17.7
287	1	0	0	0	3.0	1052	3.0	1050	0.2
288	8	0	0	0	3.4	1099	3.5	1150	- 4.4
289	0	0	0	0	2.4	1055	3.2	870	21.3
290	0	0	0	1	1.8	1168	2.5	1070	9.2
291	6	0	0	0	1.6	1122	2.2	1460	-23.2
292	1	0	0	0	2.9	1234	3.0	1650	-25.2
293	4	1	0	0	2.5	1199	2.7	1460	-17.9
294	6	0	0	0	1.9	1117	2.0	1760	-36.5
295	0	0	0	0	2.6	910	3.2	1060	-14.2
296	6	0	0	0	2.9	1010	2.9	1260	-19.8
297	9	0	0	0	2.7	1007	2.8	1170	-13.9
298	3	0	0	0	3.4	1012	3.6	1051	- 3.7
299	7	0	0	0	2.9	1033	3.0	1340	-22.9
300	9	2	0	0	2.8	1160	3.1	1640	-29.3

UF - Number of times an uneven flow condition was detected

FR - Number of times a reversal in flow was detected, maximum of one per frame

Table II. (continued)

RUN #	FILL UF	FILL FR	POST FILL UF	POST FILL FR	COMPUTER TIME	COMPUTER EVAL	VISUAL TIME	VISUAL EVAL	EVAL % DIF
319	3	0	0	0	3.1	984	2.9	1760	-44.1
320	3	0	0	0	1.5	925	1.7	1880	-50.1
321	5	0	0	0	0.8	1152	0.9	2190	-47.4
322	10	2	0	0	2.8	1200	2.7	1861	-35.5
323	8	0	0	0	1.6	971	1.4	2090	-53.5
324	4	0	0	0	0.9	977	0.8	2280	-57.1
325	8	0	0	0	3.3	940	3.0	1760	-46.6
326	4	0	0	0	1.3	927	1.4	1970	-52.9
327	9	0	0	0	0.9	1006	0.9	2100	-52.1
328	3	0	0	0	2.5	931	2.8	1960	-52.5
329	6	0	0	0	2.5	971	1.5	2360	-58.9
330	4	0	0	2	1.1	983	1.9	2610	-62.3
331	4	0	0	0	2.7	907	3.1	1960	-53.7
332	2	0	0	0	1.1	853	1.4	1890	-54.9
333	2	0	0	0	0.9	835	2.1	2380	-64.9
334	13	2	0	0	3.2	1172	3.1	2160	-45.7
335	9	0	2	2	2.0	1106	2.4	3060	-63.9
336	6	0	0	0	1.0	1136	0.9	2900	-60.8
337	4	1	0	0	3.0	1033	3.2	2170	-52.4
338	2	0	0	0	1.4	912	1.3	2070	-55.9
339	6	0	2	2	0.9	1178	0.9	2250	-47.6
359	3	0	0	0	3.0	969	3.1	850	14.0
360	6	0	0	0	2.8	971	2.6	860	12.9
361	5	0	0	0	1.7	966	1.9	1070	- 9.7
362	9	0	0	0	1.9	1525	2.4	930	64.0
363	4	0	3	2	1.2	1113	1.3	1080	3.1
364	3	0	0	0	0.8	922	0.9	1150	-19.8
365	2	0	0	0	3.3	976	3.4	1040	- 6.2
366	16	5	0	0	2.5	1639	2.5	1160	41.3
367	4	0	1	3	1.1	1034	1.9	1440	-28.2
466	10	1	0	0	3.2	924	3.0	750	23.2
467	19	4	1	0	2.2	1289	2.2	950	35.7
468	23	1	1	1	1.9	996	1.7	1450	-31.3
469A	1	0	0	0	3.0	908	3.2	730	24.4
469B	1	0	0	0	3.0	857	3.2	730	17.4

UF - Number of times an uneven flow condition was detected

FR - Number of times a reversal in flow was detected, maximum of one per frame

Table II. (continued)

RUN #	FILL UF	FILL FR	POST FILL UF	POST FILL FR	COMPUTER TIME	COMPUTER EVAL	VISUAL TIME	VISUAL EVAL	EVAL % DIF
470	1	0	0	0	2.8	805	3.0	1050	-23.3
471	4	0	0	0	2.9	787	2.4	1250	-37.0
472	1	0	0	0	2.9	849	2.9	940	- 9.7
473	8	0	0	0	2.4	856	2.3	1440	-40.6
474	20	1	0	0	1.9	1066	2.0	1460	-27.0
475	2	0	0	0	2.8	894	3.1	720	24.2
476	3	1	0	0	2.6	872	2.8	1040	-16.2
477	7	1	0	0	2.2	976	2.5	1340	-27.2
478	0	0	0	0	3.2	911	3.5	740	23.1
479	4	2	0	0	3.3	1120	3.2	1050	6.7
480	4	0	0	0	2.7	880	3.0	1130	-22.1
481	0	0	0	0	2.8	906	3.2	940	- 3.6
482	4	2	0	0	2.9	1041	3.0	950	9.6
483	5	0	0	0	2.1	821	2.2	1440	-43.0
484	2	0	0	0	2.7	845	2.2	760	11.2
485	5	0	0	0	2.6	816	2.6	940	-13.2
486	6	0	0	0	2.2	848	2.5	1060	-20.0
487	3	0	0	0	3.1	871	3.4	650	34.0
488	0	0	0	0	2.6	791	3.3	750	5.5
489	1	0	0	0	2.0	791	2.8	1230	-35.7
490	1	0	0	0	2.4	794	2.9	1030	-22.9
491	4	0	0	0	2.2	837	2.6	1470	-43.1
492	8	0	0	0	1.6	889	1.9	1520	-41.5
493	1	0	0	0	2.5	814	2.4	1240	-34.4
494	4	1	0	0	1.6	967	1.6	1120	-13.7
495	4	0	0	0	1.4	802	1.4	1550	-48.3
496	2	0	0	0	2.0	875	2.6	830	5.4
497	11	0	0	0	1.5	848	1.5	1440	-41.1
498	14	0	0	0	1.4	1012	1.4	2050	-50.6
499	1	0	0	0	2.5	768	2.9	1150	-33.2
500	5	0	0	0	2.0	810	2.1	1150	-29.6
501	3	0	0	0	1.6	766	1.7	1550	-50.6
502	0	0	0	0	3.0	773	3.6	620	24.7
503	0	0	0	0	2.6	750	3.3	750	0.0
504	8	0	0	0	2.9	799	2.6	930	-14.1
505	5	0	0	0	1.9	798	1.6	1030	-22.5
506	2	0	0	0	2.9	760	2.2	730	4.1
507	4	0	0	0	2.2	773	2.5	1120	-31.0
508	3	0	0	0	2.2	724	2.5	1240	-41.6
509	1	0	0	0	2.1	792	3.2	730	8.5
510	1	0	0	0	2.4	707	2.9	920	-23.2
511	2	0	0	0	2.1	749	2.3	1150	-34.9

108 total evaluations, avg diff. = -15.02, std = 30.38

RESULTS

The computer evaluations and visual evaluations are compared in Table II. It should be remembered that evaluation numbers are actually penalty points. A high evaluation number represents a bad fill. The lower numbers represent the better runs. Data for the visual evaluations were taken from Mark Miller's work [13] and the project lab notes [14]. This table shows the uneven fills (UF) and flow reversals (FR) in fill and post-fill, as found by the computer. These counts depend on how long it took for the reversed flow or uneven fill to occur. If a problem showed up in more than one frame, it was counted more than once. This has the effect of weighting more severely the problems that take up more time. This is at least somewhat realistic, because the longer a problem exists, the more likely it is to cause such things as early freezing in the mold. In general, if a run was poor (high evaluation number) in the visual evaluations, it also was poor in the computer rankings. There are some significant differences.

Runs 147-156 show computer rankings somewhat higher than the visual observations. In both cases, these values are close to each other, showing that the computer evaluations of turbulence were higher than the visual rankings.

Runs 280-300 show similar rankings by computer and visual evaluation. Some values were significantly lower in the computer rankings.

Runs 319-339 were significantly lower in the computer rankings.

Runs 359-367 were similar between visual and computer evaluations, except 362 and 366. Both had significant flow problems that weren't noted in the visual evaluations.

Runs 466-511 show computer rankings generally lower than the visual observations. Some were lower than the basis, run 503, despite having uneven flows. This was because the flows were smoother than run 503, as indicated by the pixel values in the fluid, which showed very little variation.

Tests were made to establish how repeatable the values obtained by computer analysis were. The significant factors were the time the script program was started, and the location of the profile lines. Other factors such as image brightness and fill time were accounted for in the analysis programs.

The scripts used to collect data must be set into operation by the click of a mouse key, operating under Windows. This starting time can vary depending on the user of the system. Table III shows the results of the start time study. The scripts used to collect the data for each run set were identical. The time bars appearing on the videotape (placed on the tape during slow-motion

Table III. Start Time Test Results

EFFECT OF START TIME ON EVALUATION NUMBER, EVAL.COM FILES CREATED AT PROGRESSIVE 1/30 SECOND REAL-TIME INTERVALS

RUN #	FILL UF	FILL FR	POST FILL UF	POST FILL FR	COMPUTER TIME	COMPUTER EVAL	VISUAL TIME	VISUAL EVAL
286	1	0	0	0	3.0	1106	4.1	940
286A	1	0	0	0	3.1	1083		
286B	0	0	0	0	3.1	1043		
286C	0	0	0	0	2.9	1074		
286D	1	0	0	0	3.0	1070		
286E	2	0	0	0	3.0	1103		
					AVERAGE	1080		
					ST DEV	23		

RUN #	FILL UF	FILL FR	POST FILL UF	POST FILL FR	COMPUTER TIME	COMPUTER EVAL	VISUAL TIME	VISUAL EVAL
336	6	0	0	0	1.0	1136	0.9	2900
336A	4	0	0	0	1.0	989		
336B	6	0	0	0	1.0	1078		
336C	7	0	0	0	1.0	1039		
336D	3	0	0	0	1.0	938		
336E	7	0	0	0	0.9	1056		
					AVERAGE	1039		
					ST DEV	69		

RUN #	FILL UF	FILL FR	POST FILL UF	POST FILL FR	COMPUTER TIME	COMPUTER EVAL	VISUAL TIME	VISUAL EVAL
360	6	0	0	0	2.8	971	2.6	860
360A	5	0	0	0	2.9	978		
360B	6	0	0	0	2.8	1001		
360C	6	0	0	0	2.8	1052		
360D	4	0	0	0	2.2	960		
360E	6	0	0	0	2.7	1019		
					AVERAGE	997		
					ST DEV	34		

UF - Number of times an uneven flow condition was detected
FR - Number of times a reversal in flow was detected, maximum of one per frame

copying) show the real time elapsed, in hours, minutes, seconds and 30ths of a second (each full frame in the standard television signal takes 1/30 second). To test the sensitivity of evaluation number to starting time, several runs were made with slightly different starting times. the main run was started when the fluid was at the top of the column. Each succeeding run (labeled with a letter in the table) was started 1/30 second later (real time). Variation in evaluation number was greatest in run 336, amounting to -198 maximum.

Tests were also made to study the sensitivity to profile line location. The columns in the fluid flow channels are about 10-20 pixels wide. If the profile lines were too far to one side, the data collected would miss most of the turbulence and bubbles in the column. Table IV shows the results of the line location tests. The same scripts were used as for the timing tests, but all runs were started with fluid at the top of the column. The profile lines were shifted left or right a few pixels, as shown in the table. Maximum variation in these tests occurred in run 286 and was +106.

Considering the maximum variations in these tests and the standard deviation values, the likely variation in a given measurement would be up to +/- 110 in a measurement of 1100. If the complete evaluation procedure is carefully done, the expected variation in a given measurement will then be about +/- 10%. This is not an exact error

Table IV. Profile Line Location Test Results

EFFECT OF PROFILE LINE LOCATION ON EVALUATION NUMBER
COLUMN OFFSET IN PIXELS, EVAL.COM FILES

RUN	OFF-SET	FILL UF	FILL FR	POST FILL UF	POST FILL FR	COMPUTER TIME	COMPUTER EVAL	VISUAL TIME	VISUAL EVAL
286	- 2	1	0	0	0	3.0	1093		
286	- 1	1	0	0	0	3.1	1045		
286	0	1	0	0	0	3.0	1106	4.1	940
286	+ 1	3	0	0	1	2.5	1212		
286	+ 2	2	0	0	1	3.1	1084		
						AVERAGE	1108		
						ST DEV	62		

RUN	OFF-SET	FILL UF	FILL FR	POST FILL UF	POST FILL FR	COMPUTER TIME	COMPUTER EVAL	VISUAL TIME	VISUAL EVAL
336	- 2	7	0	0	0	1.0	1066		
336	- 1	7	0	0	0	1.1	1048		
336	0	6	0	0	0	1.0	1136	0.9	2900
336	+ 1	7	0	0	0	1.1	1066		
336	+ 2	5	0	0	0	1.1	1089		
						AVERAGE	1081		
						ST DEV	34		

RUN	OFF-SET	FILL UF	FILL FR	POST FILL UF	POST FILL FR	COMPUTER TIME	COMPUTER EVAL	VISUAL TIME	VISUAL EVAL
360	- 2	9	0	0	0	2.3	1013		
360	- 1	7	0	0	0	2.9	988		
360	0	6	0	0	0	2.8	971	2.6	860
360	+ 1	5	0	0	0	2.9	1024		
360	+ 2	6	0	0	0	2.9	1056		
						AVERAGE	1010		
						ST DEV	33		

UF - Number of times an uneven flow condition was detected

FR - Number of times a reversal in flow was detected, maximum of one per frame

analysis, but does give an estimate of the repeatability of the computer method.

An estimate of the accuracy of the computer method is given by numerical comparison with the visual method. The percentage difference between the computer and visual methods is shown as the last column in Table II. For the 108 total evaluations done, the computer evaluations averaged 15.0% lower than the visual evaluations, with a standard deviation of 30.4%. This shows the computer method to produce results generally close to the visual method, but when differences exist they tend to be large.

The calculated fill times agree with most of the visually obtained times within 0.5 second. Those that differ substantially agree when a factor of 0.5 is added to the calculated values. Slow pours often took an extra half-second for fluid to splash through the tundish and solidly enter the downsprue. The calculated values were obtained by timing between fluid actually in the column and fluid filling the top rows. Timing was done visually by timing from fluid entering the tundish to fluid exiting the top of the mold, as shown in figure 7. A slight time difference is to be expected.

Table V lists runs where the computer evaluations were significantly different from the original visual evaluations. It contains a more detailed description of the original evaluations. The major flow problems noted visually are shown for each run. These include flow

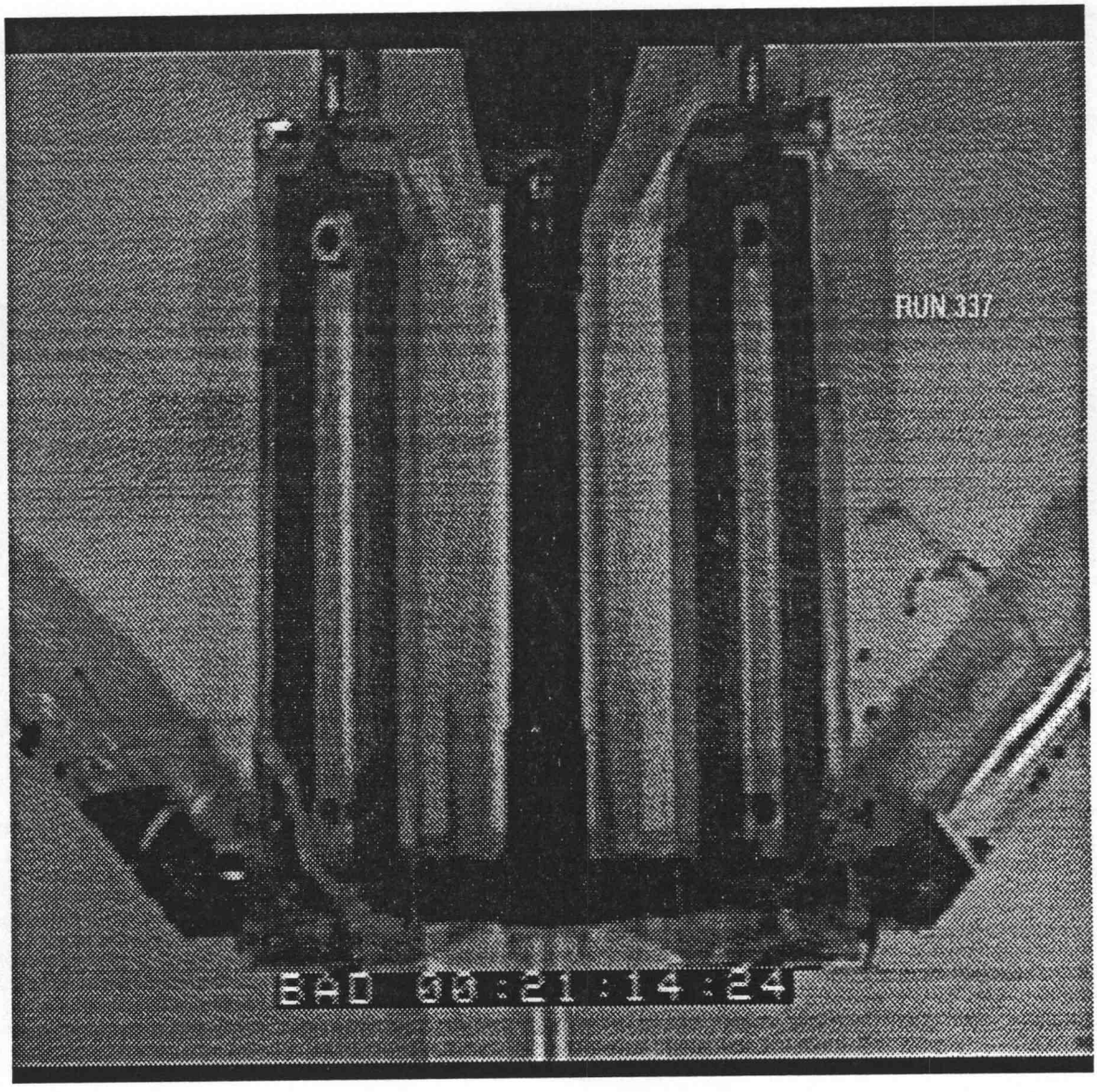

Figure 7. Completely Filled Mold

Table V. Discrepancies between computer and visual evaluations, selected runs

RUN #	COMPUTER TIME	COMPUTER EVAL	VISUAL TIME	VISUAL EVAL	PROBLEMS COUNTED VISUALLY BY RKS*
319	3.1	984	2.9	1760	13 SP, 2 FS
320	1.5	925	1.7	1880	14 SP, 2 FS
321	0.8	1152	0.9	2190	17 SP, 2 FS
322	2.8	1200	2.7	1861	14 SP, 2 FS
323	1.6	971	1.4	2090	14 SP, 4 FS
324	0.9	977	0.8	2280	16 SP, 4 FS
325	3.3	940	3.0	1760	13 SP, 2 FS
326	1.3	927	1.4	1970	15 SP, 2 FS
327	0.9	1006	0.9	2100	16 SP, 2 FS
359	3.0	969	3.1	850	7 SP
360	2.8	971	2.6	860	8 SP
361	1.7	966	1.9	1070	9 SP
362	1.9	1525	2.4	930	6 SP, 1 FS
363	1.2	1113	1.3	1080	10 SP
364	0.8	922	0.9	1150	7 SP, 1 FS
365	3.3	976	3.4	1040	6 SP, 2 FS
366	2.5	1639	2.5	1160	6 SP, 1 FS, 1 FR
367	1.1	1034	1.9	1440	9 SP, 2 FS, 1 FR
490	2.4	794	2.9	1030	9 SP
491	2.2	837	2.6	1470	9 SP, 2 FS, 1 SL
492	1.6	889	1.9	1520	11 SP, 2 FS, 1 SL
493	2.5	814	2.4	1240	10 SP
494	1.6	967	1.6	1120	9 SP
495	1.4	802	1.4	1550	12 SP
496	2.0	875	2.6	830	7 SP
497	1.5	848	1.5	1440	11 SP, 2 FS
498	1.4	1012	1.4	2050	10 SP, , 2 SL

* RKS - Russell K. Swan, evaluations used in the Mark Miller project, from project lab notes.

SP - Number of splashes counted
SL - Number of sloshes counted (splashing flow from one side of horizontal runner to other)
UF - Number of times an uneven flow condition was detected
FR - Number of times a reversal in flow was detected, maximum of one per frame
FS - Number of flow separations counted

separations (FS), splashes (SP), sloshes and flow reversals (FR). The computer method as used was not capable of detecting these problems directly, so large differences occurred between visual and computer rankings when many of these problems were present.

Table VI shows a comparison of the results obtained from the high-speed camera and the standard speed camcorder. The reduced contrast available from the high-speed camera images affects the accuracy of the evaluations. Profile lines placed on images obtained with the high-speed camera produced very flat and even graphs of pixel value versus position. Bubbles and edges appeared very similar and the background was only slightly lighter than the fluid. Evaluation numbers obtained from these images was still comparable with those obtained from the slow-motion copies of the regular videotapes and the visual evaluations.

The advantages of the high-speed camera are greatly improved resolution and the 1/17 speed reduction. Since the camera records at 500 pictures per second and the tape is replayed at 30 frames per second, a 16.67 slow-motion tape is produced without the need for copying. If a more elaborate system is developed, the high-speed camera could be put to use in situations where the contrast loss is not as significant as the gain in resolution. For this project, the contrast was more important for reliable edge detection and turbulence measurement. As table VI shows, the high-speed camera can produce usable results.

Table VI. Comparison of High-Speed and Standard Camera

RUN #	FILL UF	FILL FR	POST FILL UF	POST FILL FR	COMPUTER TIME	COMPUTER EVAL	VISUAL TIME	VISUAL EVAL
51H	14	2	0	0	3.10	1004	3.3	1200
51N	4	1	0	0	3.20	1126		
52H	6	0	0	2	1.00	723	2.0	1410
52N	4	2	0	0	2.00	1168		
53H	10	0	0	0	1.40	793	1.2	1020
53N	5	0	0	0	1.60	787		

H - Run filmed with high-speed camera
N - Run filmed with normal-speed camera

UF - Number of times an uneven flow condition was detected
FR - Number of times a reversal in flow was detected, maximum of one per frame

CONCLUSIONS

This computer analysis procedure demonstrated the possibility of using computer image processing for fluid motion studies. Secondly, it could give usable results for the mold-filling studies, within limitations. Finally, the results obtained compare well with those from visual evaluations, within those limitations.

The method can detect the bubble/turbulence level, evenness of fill, and the major heavy turbulence problem of flow reversals. It can establish a time-of fill value between two stated events in the filling process. The computer method is more reliable for determining these factors, as it applies the same criteria in every analysis.

The computer method is, however, unable to detect several other problems. Table V shows the differences resulting from inability to spot all problems. A review of the original evaluations found in Mark Miller's thesis shows that the high evaluation numbers obtained for many of the runs were due to criteria that the computer method cannot detect in its present form.

One important problem is flow separations. They occur when fluid exiting a choke or turning a corner leaves the mold surface. This allows a pocket of air to remain in the mold until the mold is nearly filled. The single profile lines in the bottom rows are unable to detect this.

The other major contributor to high evaluations is splashing. When a bubble bursts, it will sometimes throw out many small fluid particles. Fluid striking the walls of the mold also does this at times. Sometimes fluid will enter a mold without splashing, and bubbles will disappear smoothly when they reach the top of the fluid column. It is often a matter of chance. Repeated runs of the same configuration for statistical purposes were not done in the past studies, so the repeatability of these high evaluations is not certain.

Occasionally, the computer produced higher evaluations. This is because the computer is able to give a definite measure of fluid turbulence. It can quantify the large-scale mixing of the fluid and air to some extent. This results in heavier weighting of rough flows. The major differences were likely caused by the greater weighting given to reversed flows and uneven flows since they were counted more than once. It is difficult to separate multiple countings from multiple reverses, although some success was achieved by comparing only every other frame, as was done in the CALC.PAS program.

The numerical comparisons between computer and visual evaluations reflects these limitations. When the flow problems were detectable by computer, the computer produced results that agreed well with the visual evaluations. When the problems were not detectable, the computer results were often very much lower than the visual numbers.

Global Lab Image script editor memory limitations limit the method to one profile line for each region. This reduces the 2-1/2 dimensional information available in the videotape down to a single one-dimensional data set. A few vertical lines in each row, especially near the choke points and corners, a line in the downsprue and some lines at the exit region of the top runners would make the analysis complete. In its present form it illustrates the practicality of using an image processing system to do the evaluations of fluid motion studies in mold testing. The results obtained in this study do not alter the general conclusions reached earlier, based on the visual evaluations. Rather, this study adds to the information available concerning methods for conducting future research.

IMPROVEMENTS

There are several ways the system could be improved too Produce complete results. More data points from additional profile lines in the bottom rows need to be collected to detect flow separations. More frames to analyze would give better indications of the size of bubble splashes. The script editor/interpreter in GLI v1.01 is at its memory limit with the 8 profile lines currently used. To add more lines would require using a separate 'C' editor and compiler. Microsoft 'C' v6.00 is the recommended compiler, but problems with making the GLI script header files available to the Microsoft compiler prevented its use. Improved versions of GLI may overcome these difficulties. The main difficulty comes from the need to collect data while the videotape is running. More data could be collected by copying the tape to an even slower speed. This would require copying the tape twice, because the available one-inch tape equipment can only copy tapesat 1/18 speed or faster. Much more time and expense would need to be invested in gathering data. Another solution would be to improve the quality of the VCR sync signal, either through signal conditioning or the time base corrector setup mentioned earlier. This would allow analysis to be done frame by frame in pause mode. As much data as needed for a complete analysis could be taken easily then. Another potential option is a high-

speed camera. If a high enough speed and sufficient image contrast can be obtained, the extra procedures for slow-motion copying would be eliminated and enough data could be taken to give good results.

As technology improves, the cost of higher-speed frame grabbers and processors will drop. The entire process of gathering data and analyzing it could be done in real time with dedicated hardware. A lower-cost option may soon be a frame grabber with less sensitivity to the VCR sync signal. Data Translation makes another frame grabber, the QuickCapture board, for the Apple Macintosh computers. These boards work very well with the VCR in pause mode. The software required to get the data out of the board with reasonable ease was too expensive at the time the current system was assembled. The Macintosh system would be ideal for collecting a few pieces of data from every frame in a test run. Much better systems will be on the market at low cost within a few years.

When fully adequate systems are affordable, a complete, reliable and easy-to-use system will be readily assembled. One possible configuration would include the DT-3851 frame grabber and associated software. This board has improved sync handling, allowing the pause feature of the VCR to be used. With such a board it would be possible to take virtually unlimited amounts of data from each frame.

BIBLIOGRAPHY

1. Bradley, Elihu F., High Performance Castings, A Technical Guide, ASM International, Materials Park, Ohio, c1989, pp. 67-70.

2. Ibid., p 153.

3. Lade, David R.,PhD., ed., CRC Handbook of Chemistry and Physics, 73rd edition, CRC Press Inc., Ann Arbor, c1992, p 6-10.

4. Miller, Mark W., An Experimental Study of the Fluid Mechanics of Filling a Small Part Modular Mold, Oregon State University 1991, p 63.

5. Ibid., p 38.

6. Ibid., p 56.

7. Baxes, Gregory A., Digital Image Processing, A Practical Primer, Prentice-Hall, Englewood Cliffs, New Jersey, c1984, pp. 70-73.

8. Anner, George F., Elements of Television Systems, Prentice-Hall, Inc., New York, c1951, p 8.

9. Data Translation Inc., "Choosing Image Processing Boards and Software," 1993 Product Handbook, Data Translation Inc., Marlboro, Massachusetts, c1993, pp. 32-37.

10. Mood, Alexander M., "Statistics," McGraw-Hill Encyclopedia of Science and Technology, vol 17, 7th ed., McGraw-Hill, Inc., New York, c1992, pp. 354-356.

11. Baxes, Gregory A., Digital Image Processing, A Practical Primer, Prentice-Hall, Englewood Cliffs, New Jersey, c1984, pp. 52-56.

12. Baxes, Gregory A., Digital Image Processing, A Practical Primer, Prentice-Hall, Englewood Cliffs, New Jersey, c1984, p 169.

13. Miller, Mark W., An Experimental Study of the Fluid Mechanics of Filling a Small Part Modular Mold, Oregon State University 1991, pp. 40-55.

14. Swan, Russell K., assorted lab notes in OMI project files, Mechanical Engineering Department, Oregon State University, 1990-1991.

APPENDICES

APPENDIX A. EXAMPLE SCRIPT FILE

Appendix A. Example Script File

```
/* Interpreted 'C' script file for collecting pixel data */

#include <gstruct.h>
#include <scriptlb.h>
#include <fg.h>

#define v1x 168;
#define v2x 217;
#define v3x 421;
#define v4x 468;
#define h1y 391;
#define h2y 394;
#define h3y 77;
#define h4y 76;

struct fg_info finfo;
unsigned char profile_data[1024];
int profile_xlist[1024], profile_ylist[1024];
int profile_len, profile_cnt;
int profile_x1, profile_y1;
int profile_x2, profile_y2;
char instring[4];
int  i;

FILE *fp=fopen("run337.dat" , "w");

void test337 ()
{
    for (i=1; i<=13; i++)
    {
            /* preparing to take a picture, gli ch 2 */
       fg_get_info(0,&finfo); /* get frame grabber info */
       finfo.channel=1;
       finfo.ilutno=0;
       finfo.ilutno=0;
       finfo.sync_mode=1;
       finfo.live_on=0;
       finfo.average_cnt=1;
       fg_set_info(0, &finfo); /* set frame grabber info */
       fg_picture(0);          /* take picture */

       /* Preparing to collect pixel data for region V1 */
         profile_x1 = v1x;
         profile_y1 = h1y;
         profile_x2 = 2+v1x;
         profile_y2 = h3y;
         profile_len = line_coords (profile_x1, profile_y1,
                       profile_x2, profile_y2,
                       profile_xlist, profile_ylist);
```

```
for (profile_cnt = 0; profile_cnt < profile_len;
     profile_cnt++)
  {
       buffer_getln(0, profile_xlist[profile_cnt],

                profile_ylist[profile_cnt], 1,
                profile_data+profile_cnt);
       sprintf(instring,"%d\n",
                profile_data[profile_cnt]);
       fputs(instring,fp);
   }
  sprintf(instring,"-1  \n");
  fputs(instring,fp);

 /* Preparing to collect pixel data for region V2 */
   profile_x1 = v2x;
   profile_y1 = h1y;
   profile_x2 = 2+v2x;
   profile_y2 = h3y;
   profile_len = line_coords (profile_x1, profile_y1,
                 profile_x2, profile_y2,
                 profile_xlist, profile_ylist);
   for (profile_cnt = 0; profile_cnt < profile_len;
        profile_cnt++)
  {
       buffer_getln(0, profile_xlist[profile_cnt],
                    profile_ylist[profile_cnt], 1,
                    profile_data+profile_cnt);
       sprintf(instring,"%d \n",
                profile_data[profile_cnt]);
       fputs(instring,fp);
  }
  sprintf(instring,"-1  \n");
  fputs(instring,fp);

 /* Preparing to collect pixel data for region V3 */
   profile_x1 = v3x;
   profile_y1 = h2y;
   profile_x2 = v3x;
   profile_y2 = h4y;
   profile_len = line_coords (profile_x1, profile_y1,
                 profile_x2, profile_y2,
                 profile_xlist, profile_ylist);
   for (profile_cnt = 0; profile_cnt < profile_len;
        profile_cnt++)
  {
       buffer_getln(0, profile_xlist[profile_cnt],
                    profile_ylist[profile_cnt], 1,
                    profile_data+profile_cnt);
       sprintf(instring,"%d \n",
                profile_data[profile_cnt]);
```

```
        fputs(instring,fp);
    }
    sprintf(instring,"-1  \n");
    fputs(instring,fp);

/* Preparing to collect pixel data for region V4 */
    profile_x1 = v4x;
    profile_y1 = h2y;
    profile_x2 = v4x;
    profile_y2 = h4y;
    profile_len = line_coords (profile_x1, profile_y1,
                  profile_x2, profile_y2,
                  profile_xlist, profile_ylist);
    for (profile_cnt = 0; profile_cnt < profile_len;
         profile_cnt++)
    {
        buffer_getln(0, profile_xlist[profile_cnt],
                     profile_ylist[profile_cnt], 1,
                     profile_data+profile_cnt);
        sprintf(instring,"%d \n",
                profile_data[profile_cnt]);
        fputs(instring,fp);
    }
    sprintf(instring,"-1   \n");
    fputs(instring,fp);

/* Preparing to collect pixel data for region H1 */
    profile_x1 = v1x;
    profile_y1 = h1y;
    profile_x2 = v2x;
    profile_y2 = h1y;
    profile_len = line_coords (profile_x1, profile_y1,
                  profile_x2, profile_y2,
                  profile_xlist, profile_ylist);
    for (profile_cnt = 0; profile_cnt < profile_len;
         profile_cnt++)
    {
        buffer_getln(0, profile_xlist[profile_cnt],
                     profile_ylist[profile_cnt], 1,
                     profile_data+profile_cnt);
        sprintf(instring,"%d \n",
                profile_data[profile_cnt]);
        fputs(instring,fp);
    }
    sprintf(instring,"-1   \n");
    fputs(instring,fp);

/* Preparing to collect pixel data for region H2 */
    profile_x1 = v3x;
    profile_y1 = h2y;
```

```
    profile_x2 = v4x;
    profile_y2 = h2y;


profile_len = line_coords (profile_x1, profile_y1,
              profile_x2, profile_y2,
                  profile_xlist, profile_ylist);
    for (profile_cnt = 0; profile_cnt < profile_len;
         profile_cnt++)
   {
        buffer_getln(0, profile_xlist[profile_cnt],
               profile_ylist[profile_cnt], 1,
               profile_data+profile_cnt);
        sprintf(instring,"%d \n",
                profile_data[profile_cnt]);
        fputs(instring,fp);
   }
   sprintf(instring,"-1   \n");
   fputs(instring,fp);

  /* Preparing to collect pixel data for region H3 */
    profile_x1 = 2+v1x;
    profile_y1 = h3y;
    profile_x2 = 2+v2x;
    profile_y2 = h3y;
    profile_len = line_coords (profile_x1, profile_y1,
                  profile_x2, profile_y2,
                  profile_xlist, profile_ylist);
    for (profile_cnt = 0; profile_cnt < profile_len;
         profile_cnt++)
   {
        buffer_getln(0, profile_xlist[profile_cnt],
                     profile_ylist[profile_cnt], 1,
                     profile_data+profile_cnt);
        sprintf(instring,"%d \n",
                profile_data[profile_cnt]);
        fputs(instring,fp);
   }
   sprintf(instring,"-1   \n");
   fputs(instring,fp);

  /* Preparing to collect pixel data for region H4 */
    profile_x1 = v3x;
    profile_y1 = h4y;
    profile_x2 = v4x;
    profile_y2 = h4y;
    profile_len = line_coords (profile_x1, profile_y1,
                  profile_x2, profile_y2,
                  profile_xlist, profile_ylist);
```

```
          for (profile_cnt = 0; profile_cnt < profile_len;
               profile_cnt++)
          {
               buffer_getln(0, profile_xlist[profile_cnt],
                            profile_ylist[profile_cnt], 1,
                            profile_data+profile_cnt);
               sprintf(instring,"%d \n",
                       profile_data[profile_cnt]);
               fputs(instring,fp);
          }
          sprintf(instring,"-1   \n");
          fputs(instring,fp);
     }     /* end for */

        fclose(fp);
}         /* end main */

gl_script_append ("test337", test337);
```

APPENDIX B. EXAMPLE DATA FILE

Appendix B. Example Data File

Sample pixel data from profile line
Run 337 column 3 (421,391) to (422,76)
Profile line from bottom to top of column
Corresponding values here left to right, top to bottom

318pts		1	19	22	9	20	43	24	0	19	31
20	8	1	20	17	10	5	16	20	13	9	19
26	10	7	17	18	7	7	18	15	10	8	5
14	7	15	28	44	22	20	28	33	24	25	46
35	13	28	39	43	33	14	29	34	16	13	37
31	9	9	27	36	16	25	43	36	23	37	45
30	24	25	31	30	24	38	44	40	30	32	35
32	35	33	34	35	32	37	41	32	35	46	32
34	46	40	37	29	32	48	35	27	34	42	23
38	41	39	41	24	33	48	34	20	34	44	35
32	42	38	27	38	41	39	33	34	35	35	32
33	34	37	41	37	31	30	30	34	43	44	30
27	42	44	37	38	36	31	36	91	74	107	97
100	96	88	95	106	114	97	96	119	104	88	101
93	93	88	88	94	89	93	89	97	111	91	105
100	86	87	87	92	88	99	90	93	91	86	97
95	106	97	86	77	92	83	76	71	72	72	74
52	67	60	60	36	36	40	36	35	35	36	40
36	36	45	45	44	47	47	53	59	51	73	82
152	153	93	93	153	153	112	110	155	157	113	109
157	156	113	117	159	157	115	125	166	140	104	138
171	133	107	152	154	116	118	155	143	107	107	136
153	101	96	142	143	105	104	137	141	106	107	122
124	128	101	96	104	106	78	81	114	109	85	85
114	117	97	87	143	150	89	107	144	140	99	91
135	134	85	88	134	122	88	111	131	108	89	98
133	116	82	102	125	98	89	119				

APPENDIX C. LISTING OF PROGRAM 'CALC.PAS'

Appendix C. Listing of program 'CALC.PAS'

```
PROGRAM CALC;
CONST
   MAXH=101;
   MAXV=401;
   MAXF=60;
      T=9;       (* NUMBER OF VALUES TO AVERAGE *)
TYPE
   A2=ARRAY[1..4,1..MAXH] OF INTEGER;
   A3=ARRAY[1..4,1..MAXV] OF INTEGER;
   A4=ARRAY[1..MAXF,1..12,1..4] OF REAL;
            (* FRAME, RUNNER, INFO *)
            (* INFO = LENGTH, EDGE, AVG, STD *)
VAR
   VCL1,VCL2 : INTEGER;
               (* pixels before and after crack in V1 *)
   HCL1,HCL2 : INTEGER;
               (* pixels before and after crack in H1 *)

   HCUT : A2;     (* horizontal cutoff values *)
   VCUT : A3;     (* vertical cutoff values *)
   HPIX : A2;     (* pixels in rows *)
   VPIX : A3;     (* pixels in cloumns *)
   RINFO : A4;    (* data collected from RUN###.DAT files*)
   AVG,STD : REAL;
   SUM,SQSUM : REAL;
   I,J,K,L,NF : INTEGER;        (* n+1 total values *)
   CDIFF,EDGE : INTEGER;
                (* EDGE = NUMBER OF PIXELS IN FLUID *)
   HCUTMIN,VCUTMIN : INTEGER;
                (* MINIMUM ROW(H) AND  COLUMN(V) CUTOFFS *)
   CHECK1,CHECK2 : CHAR;
   FNAMES,GNAMES : STRING[13];
   F : TEXT;
   DONE1,DONE2,FOUND : BOOLEAN;

BEGIN
   DONE1:=FALSE;
   WRITELN;
   WRITELN(' DETERMINE FLOW LEVEL AND BUBBLE DENSITY ');
   WRITELN;
   WRITE(' ENTER FILE NAME (WITH EXT) : ');
   READLN(FNAMES);
   WRITELN;
   WRITELN(' READING FILE : ',FNAMES);
   WRITELN;
   ASSIGN(F,FNAMES);
   RESET(F);
```

```
L:=0;
WHILE NOT EOF(F) DO BEGIN
   L:=L+1;
   FOR I:=1 TO 4 DO BEGIN       (* VERTICAL COLUMNS *)
      J:=1;
      DONE2:=FALSE;
      WHILE NOT DONE2 DO BEGIN
         READLN(F,VPIX[I,J]);
         IF VPIX[I,J] > -1 THEN J:=J+1
            ELSE DONE2:=TRUE;
      END;
      VPIX[I,MAXV]:=J-1;    (* NUMBER OF TOTAL VALUES *)
      RINFO[L,I,1]:=J-1;

      (* CUTOFF VALUES FROM BACKGROUND *)
      IF L=1 THEN BEGIN
         (* CRACK REGION BOUNDARIES *)
         VCL1:=TRUNC(0.05*VPIX[1,MAXV]);
               (* PIXEL BEFORE CRACK IN V1 *)
         VCL2:=TRUNC(0.17*VPIX[1,MAXV]);
               (* PIXEL AFTER  CRACK IN V1 *)

         SUM:=0;
         J:=1;
         WHILE J<=VPIX[I,MAXV] DO BEGIN
            SUM:=SUM+VPIX[I,J];
            IF (I=1) AND (J>=VCL2) AND (J<VCL2+T)
            THEN BEGIN
               SUM:=SUM-VPIX[I,J-VCL2+VCL1-T+1];
               AVG:=SUM/T;
               END
            ELSE IF J>T THEN BEGIN
               SUM:=SUM-VPIX[I,J-T];
               AVG:=SUM/T;
               END
            ELSE AVG:=SUM/J;
            VCUT[I,J]:=TRUNC(AVG+0.5);
            IF (I=1) AND (J=VCL1) THEN J:=VCL2
               ELSE J:=J+1;
         END;
         SUM:=0;
         J:=1;
         WHILE J<=VPIX[I,MAXV] DO BEGIN
            SUM:=SUM+VPIX[I,J];
            IF (I=1) AND (J=VCL1) THEN J:=VCL2
               ELSE J:=J+1;
         END;
         IF I=1 THEN AVG:=SUM/(VPIX[I,MAXV]-VCL2+VCL1+1)
            ELSE AVG:=SUM/VPIX[I,MAXV];
         SQSUM:=0;
```

```
        J:=1;
        WHILE J<=VPIX[I,MAXV] DO BEGIN
          SQSUM:=SQSUM+(VPIX[I,J]-AVG)*(VPIX[I,J]-AVG);
          IF (I=1) AND (J=VCL1) THEN J:=VCL2
             ELSE J:=J+1;
        END;
        IF I=1 THEN
           STD:=SQRT(SQSUM/(VPIX[I,MAXV]-VCL2+VCL1+1))
           ELSE STD:=SQRT(SQSUM/VPIX[I,MAXV]);
        CDIFF:=TRUNC(STD/SQRT(T)+0.5);
        IF CDIFF < 5 THEN CDIFF:=5;
        WRITELN(' AVERAGE PIXEL VALUE IN COLUMN ',I:2,
                ' = ',AVG:6:4);
        WRITELN(' STANDARD DEVIATION = ',STD:6:4);
        WRITELN(' CUTOFF DIFFERENCE  = ',CDIFF:2);
        WRITELN;
        VCUTMIN:=TRUNC(AVG/2+0.5);
        J:=1;
        WHILE J<=VPIX[I,MAXV] DO BEGIN
           VCUT[I,J]:=VCUT[I,J]-CDIFF;
           IF VCUT[I,J] < VCUTMIN THEN
              VCUT[I,J]:=VCUTMIN;
           IF (I=1) AND (J=VCL1) THEN J:=VCL2
              ELSE J:=J+1;
        END;
    END;     (* IF L=1 *)
END;         (* FOR I, VERTICAL *)

FOR I:=1 TO 4 DO BEGIN       (* HORIZONTAL ROWS *)
    J:=1;
    DONE2:=FALSE;
    WHILE NOT DONE2 DO BEGIN
       READLN(F,HPIX[I,J]);
       IF HPIX[I,J] > -1 THEN J:=J+1
          ELSE DONE2:=TRUE;
    END;
    HPIX[I,MAXH]:=J-1;    (* NUMBER OF TOTAL VALUES *)
    RINFO[L,I+4,1]:=J-1;
    RINFO[L,I+8,1]:=J-1;

    (* CUTOFF VALUES FROM BACKGROUND *)
    IF L=1 THEN BEGIN
       (* CRACK REGION BOUNDARIES FOR H1 *)
    (* HCL1:=TRUNC(0.30*HPIX[1,MAXH]);    *)
       (* PIXEL BEFORE CRACK IN H1 *)
    (* HCL2:=TRUNC(0.80*HPIX[1,MAXH]);    *)
       (* PIXEL AFTER  CRACK IN H1 *)
       HCL1:=25;  (* IGNORE H1 CRACK FOR TEST *)
       HCL2:=26;  (* IGNORE H1 CRACK FOR TEST *)
```

```
SUM:=0;
J:=1;
WHILE J<=HPIX[I,MAXH] DO BEGIN
   SUM:=SUM+HPIX[I,J];
   IF (I=1) AND (J>=HCL2) AND (J<HCL2+T)
   THEN BEGIN
      SUM:=SUM-HPIX[I,J-HCL2+HCL1-T+1];
      AVG:=SUM/T;
      END
   ELSE IF J>T THEN BEGIN
      SUM:=SUM-HPIX[I,J-T];
      AVG:=SUM/T;
      END
   ELSE AVG:=SUM/J;
   HCUT[I,J]:=TRUNC(AVG+0.5);
   IF (I=1) AND (J=HCL1) THEN J:=HCL2
      ELSE J:=J+1;
END;
SUM:=0;
J:=1;
WHILE J<=HPIX[I,MAXH] DO BEGIN
   SUM:=SUM+HPIX[I,J];
   IF (I=1) AND (J=HCL1) THEN J:=HCL2
      ELSE J:=J+1;
END;
IF I=1 THEN AVG:=SUM/(HPIX[I,MAXH]-HCL2+HCL1+1)
   ELSE AVG:=SUM/HPIX[I,MAXH];
SQSUM:=0;
J:=1;
WHILE J<=HPIX[I,MAXH] DO BEGIN
  SQSUM:=SQSUM+(HPIX[I,J]-AVG)*(HPIX[I,J]-AVG);
  IF (I=1) AND (J=HCL1) THEN J:=HCL2
     ELSE J:=J+1;
END;
IF I=1 THEN
   STD:=SQRT(SQSUM/(HPIX[I,MAXH]-HCL2+HCL1+1))
   ELSE STD:=SQRT(SQSUM/HPIX[I,MAXH]);
CDIFF:=TRUNC(STD/SQRT(T)+0.5);
IF CDIFF < 5 THEN CDIFF:=5;
WRITELN(' AVERAGE PIXEL VALUE IN ROW ',I:2,
        ' = ',AVG:6:4);
WRITELN(' STANDARD DEVIATION = ',STD:6:4);
WRITELN(' CUTOFF DIFFERENCE  = ',CDIFF:2);
WRITELN;
HCUTMIN:=TRUNC(AVG/2+0.5);
J:=1;
WHILE J<=HPIX[I,MAXH] DO BEGIN
   HCUT[I,J]:=HCUT[I,J]-CDIFF;
   IF HCUT[I,J] < HCUTMIN THEN
      HCUT[I,J]:=HCUTMIN;
```

```
            IF (I=1) AND (J=HCL1) THEN J:=HCL2
               ELSE J:=J+1;
         END;
      END;     (* IF L=1 *)
   END;        (* FOR I, HORIZONTAL *)

   FOR I:=1 TO 4 DO BEGIN
      (* WRITELN(' VERTICAL REGION ',I:1,
                 '  FRAME ',L:1);  *)
      (* WRITELN('   J      PIXEL      SUM      SQSUM',
                 '      AVG          STD');     *)
      (* WRITELN('********************************', *)
            (* '************************'); *)
      SUM:=0;
      EDGE:=0;
      J:=1;
      FOUND:=FALSE;
      WHILE (J<=VPIX[I,MAXV]) AND NOT FOUND DO BEGIN
            (* CALCULATE AVERAGE PIXEL VALUE IN FLUID *)
         SUM:=SUM+VPIX[I,J];
         IF (I=1) AND (J>=VCL2) AND (J<VCL2+T) THEN
         BEGIN
            SUM:=SUM-VPIX[I,J-VCL2+VCL1-T+1];
            AVG:=SUM/T;
            END
         ELSE IF J>T THEN BEGIN
            SUM:=SUM-VPIX[I,J-T];
            AVG:=SUM/T;
            END
         ELSE AVG:=SUM/J;

            (* CALCULATE STANDARD DEVIATION OF *)
            (* PIXEL VALUES *)
         IF (I=1) AND (J>=VCL2) AND (J<VCL2+T) THEN
         BEGIN
            SQSUM:=0;
            FOR K:=(J-T-VCL2+VCL1+2) TO VCL1 DO
               SQSUM:=SQSUM+
                  (VPIX[I,K]-AVG)*(VPIX[I,K]-AVG);
            FOR K:=VCL2 TO J DO
               SQSUM:=SQSUM+
                  (VPIX[I,K]-AVG)*(VPIX[I,K]-AVG);
            STD:=SQRT(SQSUM/T);
            END
         ELSE IF J>T THEN BEGIN
            SQSUM:=0;
            FOR K:=(J-T+1) TO J DO
               SQSUM:=SQSUM+
                  (VPIX[I,K]-AVG)*(VPIX[I,K]-AVG);
            STD:=SQRT(SQSUM/T);
            END
```

```
      ELSE IF J>0 THEN BEGIN
         SQSUM:=0;
         FOR K:=1 TO J DO
            SQSUM:=SQSUM+
               (VPIX[I,K]-AVG)*(VPIX[I,K]-AVG);
         STD:=SQRT(SQSUM/J);
         END
      ELSE BEGIN
         SQSUM:=0;
         STD:=0;
      END;
      IF (AVG>=VCUT[I,J]) AND NOT FOUND THEN BEGIN
         IF J>VPIX[I,MAXV]-T THEN EDGE:=VPIX[I,MAXV]
            ELSE IF J<T THEN EDGE:=0
            ELSE EDGE:=J-T;
         FOUND:=TRUE;
      END;
   (* WRITELN(J:4,'  ',VPIX[I,J]:8,'  ',SUM:8:2, *)
   (* '  ',SQSUM:8:2,'  ',AVG:8:4,'  ',STD:8:4); *)
      IF NOT FOUND THEN
         IF (I=1) AND (J=VCL1) THEN BEGIN
            (* CRACK IN V1 *)
            J:=VCL2;
            EDGE:=VCL2-1;
            END
         ELSE BEGIN
            EDGE:=EDGE+1;
            J:=J+1;
         END;
   END;  (* WHILE J *)
   RINFO[L,I,2]:=EDGE;
(* WRITELN;                                  *)
(* WRITELN(' EDGE AT J=',EDGE:2);            *)
   SUM:=0;
   J:=1;
   WHILE J <= EDGE DO BEGIN
      SUM:=SUM+VPIX[I,J];
      IF (I=1) AND (J=VCL1) THEN J:=VCL2
         (* CRACK IN V1 *)
         ELSE J:=J+1;
   END;
   IF (I=1) AND (EDGE>=VCL2) THEN
      AVG:=SUM/(EDGE-VCL2+VCL1+1)   (* V1 *)
      ELSE IF EDGE > 0 THEN AVG:=SUM/EDGE
      ELSE AVG:=0;
   SQSUM:=0;
   J:=1;
   WHILE J <= EDGE DO BEGIN
      SQSUM:=SQSUM+(VPIX[I,J]-AVG)*(VPIX[I,J]-AVG);
      IF (I=1) AND (J=VCL1) THEN J:=VCL2
```

```
            (* CRACK IN V1 *)
            ELSE J:=J+1;
      END;
      IF (I=1) AND (EDGE>=VCL2) THEN
         STD:=SQRT(SQSUM/(EDGE-VCL2+VCL1+1))
         ELSE IF EDGE > 0 THEN STD:=SQRT(SQSUM/EDGE)
         ELSE STD:=0;
(* WRITELN(' AVERAGE PIXEL VALUE IN FLUID = ', *)
(* AVG:6:4);       *)
(* WRITELN(' STANDARD DEVIATION = ',STD:6:4); *)
(* WRITELN;     *)
      RINFO[L,I,3]:=AVG;
      RINFO[L,I,4]:=STD;
END;    (* FOR I, VERTICAL *)
FOR I:=1 TO 4 DO BEGIN
      (* WRITELN('     HORIZONTAL REGION ',I:1,          *)
            (*    '  FRAME ',L:1,'  LEFT TO RIGHT '); *)
      (* WRITELN('    J      PIXEL      SUM      SQSUM', *)
            (*    '      AVG          STD');      *)
      (*    WRITELN('*****************************', *)
               (*    '*************************');  *)
      SUM:=0;
      EDGE:=0;
      J:=1;
      FOUND:=FALSE;
      WHILE (J<=HPIX[I,MAXH]) AND NOT FOUND DO BEGIN
            (* CALCULATE AVERAGE PIXEL VALUE IN FLUID *)
         SUM:=SUM+HPIX[I,J];
         IF (I=1) AND (J>=HCL2) AND (J<HCL2+T) THEN
         BEGIN
            SUM:=SUM-HPIX[I,J-HCL2+HCL1-T+1];
            AVG:=SUM/T;
            END
         ELSE IF J>T THEN BEGIN
            SUM:=SUM-HPIX[I,J-T];
            AVG:=SUM/T;
            END
         ELSE AVG:=SUM/J;

            (* CALCULATE STANDARD DEVIATION OF *)
            (* PIXEL VALUES *)
         IF (I=1) AND (J>=HCL2) AND (J<HCL2+T) THEN
         BEGIN
            SQSUM:=0;
            FOR K:=(J-T-HCL2+HCL1+2) TO HCL1 DO
               SQSUM:=SQSUM+
                  (HPIX[I,K]-AVG)*(HPIX[I,K]-AVG);
            FOR K:=HCL2 TO J DO
               SQSUM:=SQSUM+
                  (HPIX[I,K]-AVG)*(HPIX[I,K]-AVG);
```

```
            STD:=SQRT(SQSUM/T);
            END
         ELSE IF J>T THEN BEGIN
            SQSUM:=0;
            FOR K:=(J-T+1) TO J DO
               SQSUM:=SQSUM+
                  (HPIX[I,K]-AVG)*(HPIX[I,K]-AVG);
            STD:=SQRT(SQSUM/T);
            END
         ELSE IF J>0 THEN BEGIN
            SQSUM:=0;
            FOR K:=1 TO J DO
               SQSUM:=SQSUM+
                  (HPIX[I,K]-AVG)*(HPIX[I,K]-AVG);
            STD:=SQRT(SQSUM/J);
            END
         ELSE BEGIN
            SQSUM:=0;
            STD:=0;
         END;

         IF (AVG>=HCUT[I,J]) AND NOT FOUND THEN BEGIN
            IF J > HPIX[I,MAXH]-T THEN
            EDGE:=HPIX[I,MAXH]
               ELSE IF J < T THEN EDGE:=0
               ELSE EDGE:=J-T;
            FOUND:=TRUE;
         END;
      (* WRITELN(J:4,'  ',HPIX[I,J]:8,'  ',SUM:8:2,  *)
      (*  '  ',SQSUM:8:2,'  ',AVG:8:4,'  ',STD:8:4); *)
         IF NOT FOUND THEN
            IF (I=1) AND (J=HCL1) THEN BEGIN
               (* CRACK IN H1 *)
               J:=HCL2;
               EDGE:=HCL2-1;
               END
            ELSE BEGIN
               EDGE:=EDGE+1;
               J:=J+1;
            END;
      END;  (* WHILE J *)
      RINFO[L,I+4,2]:=EDGE;
   (* WRITELN;                                  *)
   (* WRITELN(' EDGE AT J=',EDGE:2);            *)
      SUM:=0;
      J:=1;
      WHILE J <= EDGE DO BEGIN
         SUM:=SUM+HPIX[I,J];
         IF (I=1) AND (J=HCL1) THEN J:=HCL2
            (* CRACK IN H1 *)
```

```
            ELSE J:=J+1;
      END;
      IF (I=1) AND (EDGE>=HCL2) THEN
      AVG:=SUM/(EDGE-HCL2+HCL1+1)   (* H1 *)
         ELSE IF EDGE > 0 THEN AVG:=SUM/EDGE
         ELSE AVG:=0;
      SQSUM:=0;
      J:=1;
      WHILE J <= EDGE DO BEGIN
         SQSUM:=SQSUM+(HPIX[I,J]-AVG)*(HPIX[I,J]-AVG);
         IF (I=1) AND (J=HCL1) THEN J:=HCL2
            (* CRACK IN H1 *)
            ELSE J:=J+1;
      END;
      IF (I=1) AND (EDGE>=HCL2) THEN
      STD:=SQRT(SQSUM/(EDGE-HCL2+HCL1+1))
         ELSE IF EDGE > 0 THEN STD:=SQRT(SQSUM/EDGE)
         ELSE STD:=0;
(* WRITELN(' AVERAGE PIXEL VALUE IN FLUID = ', *)
         (*  AVG:6:4);       *)
(* WRITELN(' STANDARD DEVIATION = ',STD:6:4);  *)
(* WRITELN;    *)
      RINFO[L,I+4,3]:=AVG;
      RINFO[L,I+4,4]:=STD;
(* WRITELN('     HORIZONTAL REGION ',I:1, *)
         (* '  FRAME ',L:1,'  RIGHT TO LEFT '); *)
(* WRITELN('    J      PIXEL      SUM      SQSUM', *)
         (* '      AVG          STD');      *)
(* WRITELN('******************************',
         (* '******************************');  *)
      SUM:=0;
      EDGE:=HPIX[I,MAXH]+1;
      J:=HPIX[I,MAXH];
      FOUND:=FALSE;
      WHILE (J>=1) AND NOT FOUND DO BEGIN
            (* CALCULATE AVERAGE PIXEL VALUE IN FLUID *)
         SUM:=SUM+HPIX[I,J];
         IF J > HPIX[I,MAXH]-T THEN
            AVG:=SUM/(HPIX[I,MAXH]-J+1)
         ELSE IF (I=1) AND (J<=HCL1) AND (J>HCL1-T+1)
         THEN BEGIN
            SUM:=SUM-HPIX[I,J+HCL2-HCL1+T];
            AVG:=SUM/T;
            END
         ELSE BEGIN
            SUM:=SUM-HPIX[I,J+T];
            AVG:=SUM/T;
         END;

            (* CALCULATE STANDARD DEVIATION OF *)
            (* PIXEL VALUES *)
```

```
    IF (I=1) AND (J<=HCL1) AND (J>HCL1-T+1) THEN
    BEGIN
       SQSUM:=0;
       FOR K:=J TO HCL1 DO
          SQSUM:=SQSUM+
             (HPIX[I,K]-AVG)*(HPIX[I,K]-AVG);
       FOR K:=HCL2 TO (J+T+HCL2-HCL1) DO
          SQSUM:=SQSUM+
             (HPIX[I,K]-AVG)*(HPIX[I,K]-AVG);
       STD:=SQRT(SQSUM/T);
       END
    ELSE IF J<=(HPIX[I,MAXH]-T) THEN BEGIN
       SQSUM:=0;
       FOR K:=J TO (J+T-1) DO
          SQSUM:=SQSUM+
             (HPIX[I,K]-AVG)*(HPIX[I,K]-AVG);
       STD:=SQRT(SQSUM/T);
       END
    ELSE IF J<HPIX[I,MAXH] THEN BEGIN
       SQSUM:=0;
       FOR K:=J TO HPIX[I,MAXH] DO
          SQSUM:=SQSUM+
             (HPIX[I,K]-AVG)*(HPIX[I,K]-AVG);
       STD:=SQRT(SQSUM/(HPIX[I,MAXH]-J+1));
       END
    ELSE BEGIN
       SQSUM:=0;
       STD:=0;
    END;
    IF (AVG>=HCUT[I,J]) AND NOT FOUND THEN BEGIN
       IF J > HPIX[I,MAXH]-T THEN
       EDGE:=HPIX[I,MAXH]+1
          ELSE IF J<T THEN EDGE:=1
          ELSE EDGE:=J+T;
       FOUND:=TRUE;
    END;
(* WRITELN(J:4,'  ',HPIX[I,J]:8,'  ',SUM:8:2, *)
(* '  ',SQSUM:8:2,'  ',AVG:8:4,'  ',STD:8:4); *)
    IF NOT FOUND THEN
       IF (I=1) AND (J=HCL2) THEN BEGIN
          (* CRACK IN H1 *)
          J:=HCL1;
          EDGE:=HCL1+1;
          END
       ELSE BEGIN
          EDGE:=EDGE-1;
          J:=J-1;
       END;
END;  (* WHILE J *)
RINFO[L,I+8,2]:=EDGE;
```

```
    (* WRITELN;                                    *)
    (* WRITELN(' EDGE AT J=',EDGE:2);             *)
       SUM:=0;
       J:=HPIX[I,MAXH];
       WHILE J>=EDGE DO BEGIN
          SUM:=SUM+HPIX[I,J];
          IF (I=1) AND (J=HCL2) THEN J:=HCL1
             (* CRACK IN H1 *)
             ELSE J:=J-1;
       END;
       IF (I=1) AND (EDGE <= HCL1) THEN (* CRACK IN H1 *)
          AVG:=SUM/(HPIX[I,MAXH]-EDGE-HCL2+HCL1+2)
          ELSE IF EDGE<HPIX[I,MAXH] THEN
             AVG:=SUM/(HPIX[I,MAXH]-EDGE+1)
          ELSE AVG:=0;
       SQSUM:=0;
       J:=HPIX[I,MAXH];
       WHILE J >= EDGE DO BEGIN
          SQSUM:=SQSUM+(HPIX[I,J]-AVG)*(HPIX[I,J]-AVG);
          IF (I=1) AND (J=HCL2) THEN J:=HCL1
             (* CRACK IN H1 *)
             ELSE J:=J-1;
       END;
       IF (I=1) AND (EDGE <= HCL1) THEN (* CRACK IN H1 *)
         STD:=SQRT(SQSUM/(HPIX[I,MAXH]-EDGE-HCL2+HCL1+2))
         ELSE IF EDGE < HPIX[I,MAXH] THEN
            STD:=SQRT(SQSUM/(HPIX[I,MAXH]-EDGE+1))
         ELSE STD:=0;
    (* WRITELN(' AVERAGE PIXEL VALUE IN FLUID = ', *)
          (* AVG:6:4);       *)
    (* WRITELN(' STANDARD DEVIATION = ',STD:6:4);  *)
    (* WRITELN;  *)
       RINFO[L,I+8,3]:=AVG;
       RINFO[L,I+8,4]:=STD;
   END;  (* FOR I *)
END;   (* WHILE NOT EOF *)
CLOSE(F);
NF:=L;
WRITELN;
WRITELN(FNAMES,'  RINFO ',NF:4,'  TOTAL FRAMES ');
WRITELN;
WRITELN('                                        ',
        'FLUID FILLED REGION ');
WRITELN(' FRAME     RUNNER    LENGTH     ',
        'EDGE     AVG     STD ');
WRITELN('*******************************',
        '**********************');
FOR L:=1 TO NF DO
   FOR I:=1 TO 12 DO
```

```
        WRITELN(L:4,'     ',I:9,'  ',RINFO[L,I,1]:9:0,
                '   ',RINFO[L,I,2]:6:0,'  ',
                RINFO[L,I,3]:6:2,' ',RINFO[L,I,4]:6:2);
    WRITELN;
    CHECK1:='N';
    WRITE('  SAVE FRAME DATA TO FILE (Y OR N)? ');
    READLN(CHECK1);
    WRITELN;
    IF (CHECK1='Y') OR (CHECK1='y') THEN BEGIN
       GNAMES:='RINFO'+COPY(FNAMES,4,7);
       CHECK2:='N';
       WRITE('  SAVE AS ',GNAMES,' ?'); READLN(CHECK2);
       IF (CHECK2<>'Y') AND (CHECK2<>'y') THEN BEGIN
          WRITE('  ENTER FILE NAME (WITH EXT) : ');
          READLN(GNAMES);
       END;
       WRITELN;
       ASSIGN(F,GNAMES);
       REWRITE(F);
       WRITELN(F,FNAMES,'  RINFO ',NF:4,'  TOTAL FRAMES ');
       WRITELN(F);
       WRITELN(F,'         CUTOFF VALUES ');
       WRITELN(F,'  REGION   VERTCAL   HORIZONTAL ');
       WRITELN(F,'            (J=100)      (J=25)   ');
       WRITELN(F,'  ***************************** ');
       FOR I:=1 TO 4 DO
         WRITELN(F,'     ',I:2,'        ',VCUT[I,100]:2,
                 '         ' ,HCUT[I,25]:2);
       WRITELN(F);
       WRITELN(F,'                                    ',
                 'FLUID FILLED REGION ');
       WRITELN(F,' FRAME     RUNNER    LENGTH     ',
                 'EDGE     AVG    STD ');
       WRITELN(F,'*******************************',
                 '**********************');
       FOR L:=1 TO NF DO  BEGIN
          FOR I:=1 TO 12 DO
             WRITELN(F,L:4,'     ',I:9,'  ',
               RINFO[L,I,1]:9:0,'  ',RINFO[L,I,2]:6:0,
               '  ',RINFO[L,I,3]:6:2,' ',RINFO[L,I,4]:6:2);
          WRITELN(F);
       END;
       CLOSE(F);
    END;
END.
```

APPENDIX D. LISTING OF PROGRAM 'EVAL.PAS'

Appendix D. Listing of Program 'EVAL.PAS'

```
PROGRAM EVAL;
CONST
  MAXF=60;
  TPF=0.10;       (* TIME PER FRAME, SECONDS   1.80/18 *)
  W1=10;          (* UNEVEN FILL WEIGHTING FACTOR *)
  W2=100;         (* FLOW REVERSAL WEIGHTING FACTOR *)
  W3=8.762;       (* BUBBLE/TURBULENCE WEIGHTING FACTOR *)
  W5=0.10;        (* POST FILL WEIGHTING FACTOR *)
TYPE
  A4=ARRAY[1..MAXF,1..12,1..4] OF REAL;
           (* FRAME, RUNNER, INFO *)
           (* INFO = LENGTH OF COLUMN/ROW, EDGE, *)
           (* FLUID PIX AVG,STD *)
VAR
   I,J,K,L : INTEGER;      (* L=FRAME, I=RUNNER *)
   ENF,ENP : REAL;         (* EVALUATION: FILL, POST FILL *)
   EN : INTEGER;           (* EVALUATION NUMBER *)
   UFD : INTEGER;          (* DISTANCE FOR UNEVEN FILLS, *)
                           (* = 1/12 COL. LENGTH *)
   RFF,RFP,RFC : INTEGER;   (* NUMBER OF FLOW REVERSALS, *)
                               (* LOOP COUNTER *)
   UFF,UFP,UFC : INTEGER;   (* NUMBER OF UNEVEN FILLS, *)
                               (* LOOP COUNTER *)
   FF,SF : INTEGER;      (* FRAMES TO FILL, FIRST NONEMPTY *)
                            (* FRAME *)
   NF : INTEGER;         (* TOTAL FRAMES READ *)
   BTV,BTH,FT : REAL;    (* BUBBLE/TURBULENCE COUNTERS, *)
                            (* FILL TIME   *)
   W4 : REAL;            (* LENGTH FACTOR, 2/3 OF ROW  *)
                            (* LENGTH/COLUMN LENGTH *)
   RINFO : A4;
   FNAMES : STRING[13];
   GNAMES : STRING[13];
   F,G : TEXT;
   DONE,FOUND,FINI : BOOLEAN;
BEGIN
   WRITELN;
   WRITELN('    DETERMINE EVALUATION NUMBER FROM RINFO ');
   WRITELN('       DATA GENERATED BY CALC PROGRAM ');
   WRITELN('    STANDARD DEVIATION WEIGHTED, STD/AVG ');
   WRITELN;
   FINI:=FALSE;
   WRITELN(' ENTER FILE NAME TO SAVE CUMULATIVE RESULTS',
           ' (WITH EXTENSION) ');
   WRITE('   ENTER "DONE" TO END : '); READLN(GNAMES);
   WRITELN;
   IF (GNAMES='DONE') OR (GNAMES='done') THEN FINI:=TRUE;
```

```
ASSIGN(G,GNAMES);
REWRITE(G);
WRITELN(G);
WRITELN(G,'     DATA          |  FILL  | POST FILL ',
        '| FILL | EVAL ');
WRITELN(G,'     FILE          | UF  RF |  UF   RF  ',
        '| TIME |      ');
WRITELN(G,'  ------------------------------------',
        '--------------');
WHILE NOT FINI DO BEGIN
   WRITE(' ENTER RINFO FILE NAME (WITH EXT) : ');
   READLN(FNAMES);
   WRITELN;
   IF (FNAMES<>'DONE') AND (FNAMES<>'done') THEN BEGIN
      ASSIGN(F,FNAMES);
      RESET(F);
      FOR J:=1 TO 14 DO
         READLN(F);
      L:=0;
      FOUND:=FALSE;
      WHILE NOT EOF(F) DO BEGIN
         L:=L+1;
      (* READ IN RINFO VALUES, J,K DUMMY VALUES HERE *)
         FOR I:=1 TO 12 DO
            READLN(F,J,K,RINFO[L,I,1],RINFO[L,I,2],
                   RINFO[L,I,3],RINFO[L,I,4]);
         READLN(F);
         IF (RINFO[L,9,2]<=RINFO[L,9,1]) AND
            (RINFO[L,6,2]>0)
            AND NOT FOUND THEN BEGIN
               FOUND:=TRUE;
               SF:=L;
         END;
      END;
      CLOSE(F);
      NF:=L;
      IF SF>1 THEN SF:=SF-1;
      UFD:=TRUNC((RINFO[1,1,1]+RINFO[1,3,1])/24+0.5);
      (* W4:=(RINFO[1,5,1]+RINFO[1,6,1])/  *)
      (*(RINFO[1,1,1]+RINFO[1,3,1]); *)

      W4:=0.5; (* WEIGHTING EQUAL FOR BTV, BTH, *)
               (* 0.5 FOR DOUBLE COUNTING *)

      UFF:=0; RFF:=0;
      BTV:=0; BTH:=0;
      L:=SF;
      DONE:=FALSE;
      WHILE NOT DONE DO BEGIN
            (* EVENNESS OF FILL *)
```

```
      UFC:=0;
      IF ABS(RINFO[L,1,2]-RINFO[L,2,2]) > UFD
         THEN UFC:=UFC+1;
      IF ABS(RINFO[L,1,2]-RINFO[L,3,2]) > UFD
         THEN UFC:=UFC+1;
      IF ABS(RINFO[L,1,2]-RINFO[L,4,2]) > UFD
         THEN UFC:=UFC+1;
      IF ABS(RINFO[L,2,2]-RINFO[L,3,2]) > UFD
         THEN UFC:=UFC+1;
      IF ABS(RINFO[L,2,2]-RINFO[L,4,2]) > UFD
         THEN UFC:=UFC+1;
      IF ABS(RINFO[L,3,2]-RINFO[L,4,2]) > UFD
         THEN UFC:=UFC+1;
      IF UFC=3 THEN UFF:=UFF+1
         ELSE IF UFC=4 THEN UFF:=UFF+2
         ELSE IF UFC>4 THEN UFF:=UFF+3;

         (* FLOW REVERSALS *)
      IF L>2 THEN BEGIN
         RFC:=0;
         FOR I:=1 TO 4 DO
            IF ((RINFO[L,I,2]) <
               (RINFO[L-2,I,2]-UFD/3))
               THEN RFC:=RFC+1;
         IF RFC>=1 THEN RFF:=RFF+1;
      END;

         (* STD FOR BUBBLES, TURBULENCE *)
      FOR I:=1 TO 4 DO
         BTV:=BTV+RINFO[L,I,4];
      FOR I:=5 TO 6 DO
         BTH:=BTH+RINFO[L,I,4];
      IF RINFO[L,1,2]=RINFO[L,1,1]
         THEN BTH:=BTH+RINFO[L,7,4];
      IF RINFO[L,3,2]=RINFO[L,3,1]
         THEN BTH:=BTH+RINFO[L,8,4];
      FOR I:=9 TO 10 DO
         BTH:=BTH+RINFO[L,I,4];
      IF RINFO[L,2,2]=RINFO[L,2,1]
         THEN BTH:=BTH+RINFO[L,11,4];
      IF RINFO[L,4,2]=RINFO[L,4,1]
         THEN BTH:=BTH+RINFO[L,12,4];

         (* FILL TIME *)
      IF ((RINFO[L,7,2]=RINFO[L,7,1]) AND
         (RINFO[L,11,2]=1) AND
         (RINFO[L,8,2]=RINFO[L,8,1]) AND
         (RINFO[L,12,2]=1)) OR (L=NF)
         THEN DONE:=TRUE
      ELSE L:=L+1;
   END;
```

```
FF:=L;
FT:=FF*TPF;
ENF:=W1*UFF+W2*RFF+(W3*(BTV+W4*BTH))/(FF-SF+1);
WRITELN(' NUMBER OF UNEVEN FILLS = ',UFF:2);
WRITELN(' NUMBER OF FLOW REVERSALS = ',RFF:2);
WRITELN(' W4 = ',W4:4:3);
WRITELN(' BTV = ',(BTV/(FF-SF+1)):4:3,
        '   BTH = ',(BTH/(FF-SF+1)):4:3);
WRITELN(' EVALUATION NUMBER, FILL = ',ENF:4:3);
WRITELN;
   (* POST FILL *)
UFP:=0;  RFP:=0;
BTV:=0; BTH:=0;
L:=L+1;
IF L>NF THEN DONE:=TRUE ELSE DONE:=FALSE;
WHILE NOT DONE DO BEGIN
      (* EVENNESS OF FILL *)
   UFC:=0;
   IF ABS(RINFO[L,1,2]-RINFO[L,2,2]) > UFD
      THEN UFC:=UFC+1;
   IF ABS(RINFO[L,1,2]-RINFO[L,3,2]) > UFD
      THEN UFC:=UFC+1;
   IF ABS(RINFO[L,1,2]-RINFO[L,4,2]) > UFD
      THEN UFC:=UFC+1;
   IF ABS(RINFO[L,2,2]-RINFO[L,3,2]) > UFD
      THEN UFC:=UFC+1;
   IF ABS(RINFO[L,2,2]-RINFO[L,4,2]) > UFD
      THEN UFC:=UFC+1;
   IF ABS(RINFO[L,3,2]-RINFO[L,4,2]) > UFD
      THEN UFC:=UFC+1;
   IF UFC=3 THEN UFP:=UFP+1
      ELSE IF UFC=4 THEN UFP:=UFP+2
      ELSE IF UFC>4 THEN UFP:=UFP+3;

      (* FLOW REVERSALS *)

   IF L>2 THEN BEGIN
      RFC:=0;
      FOR I:=1 TO 4 DO
         IF ((RINFO[L,I,2]) <
            (RINFO[L-2,I,2]-UFD/3))
            THEN RFC:=RFC+1;
      IF RFC>=1 THEN RFP:=RFP+1;
   END;

      (* STD FOR BUBBLES, TURBULENCE *)
   FOR I:=1 TO 4 DO
      BTV:=BTV+RINFO[L,I,4];
   FOR I:=5 TO 6 DO
      BTH:=BTH+RINFO[L,I,4];
```

```
            IF RINFO[L,1,2]=RINFO[L,1,1]
               THEN BTH:=BTH+RINFO[L,7,4];
            IF RINFO[L,3,2]=RINFO[L,3,1]
               THEN BTH:=BTH+RINFO[L,8,4];
            FOR I:=9 TO 10 DO
               BTH:=BTH+RINFO[L,I,4];
            IF RINFO[L,2,2]=RINFO[L,2,1]
               THEN BTH:=BTH+RINFO[L,11,4];
            IF RINFO[L,4,2]=RINFO[L,4,1]
               THEN BTH:=BTH+RINFO[L,12,4];
            IF (L=NF) THEN DONE:=TRUE
               ELSE L:=L+1;
         END;
         ENP:=0;
         IF L>FF THEN BEGIN
            ENP:=W1*UFP+W2*RFP+(W3*(BTV+W4*BTH))/(L-FF);
            WRITELN(' FILL TIME = ',FT:4:2);
            WRITELN(' START FRAME = ',SF:2);
            WRITELN(' FRAMES TO FILL = ',FF:2);
            WRITELN(' FRAMES POST FILL = ',(L-FF):2);
            WRITELN;
            WRITELN(' NUMBER OF UNEVEN FILLS,',
                    ' POST = ',UFP:2);
            WRITELN(' NUMBER OF FLOW REVERSALS,',
                    ' POST = ',RFP:2);
            WRITELN(' BTV = ',(BTV/(L-FF)):4:3,
                    '   BTH = ',(BTH/(L-FF)):4:3);
            WRITELN(' EVALUATION NUMBER, POST = ',ENP:4:3);
            WRITELN;
         END;
         EN:=TRUNC(ENF+W5*ENP+0.5);
         WRITELN(' EVALUATION NUMBER = ',EN:2);
         WRITELN;
         WRITELN(G,'    ',FNAMES,'  |',UFF:3,RFF:4,' |',
                   UFP:4,RFP:5,'  |',FT:5:2,' |',EN:5);
         END
      ELSE FINI:=TRUE;
   END;
   WRITELN(G,'  ------------------------------------',
             '--------------');
   CLOSE(G);
END.
```

APPENDIX E. LISTING OF PROGRAM 'RENUM.PAS'

Appendix E. Listing of Program 'RENUM.PAS'

```
PROGRAM RENUM;   (* EDIT OUT TIME BAR LINES IN *)
                 (* RUN###.DAT FILES *)
CONST
   MAX=401;
TYPE
   A1=ARRAY[1..MAX] OF INTEGER;
VAR
   VBL1,VBL2 : INTEGER;   (* PIXELS BEFORE, AFTER *)
                             (* BLACK AREA *)
   I,J : INTEGER;          (* REGION, SUBSCRIPT COUNTERS *)
   PIX : A1;               (* PIXEL VALUE READ FROM FILE *)
   FNAMES,GNAMES : STRING[13];    (* INCOMING, OUTGOING *)
                                  (* FILE NAMES *)
   F,G : TEXT;
   DONE : BOOLEAN;
BEGIN
   WRITELN;
   WRITELN(' RE-WRITE RUN###.DAT FILES TO EDIT TIME BAR
LINES ');
   WRITELN;
   WRITE(' ENTER RUN FILE NAME (WITH EXTENSION) : ');
READLN(FNAMES);
   WRITELN;
   GNAMES:='RMD'+COPY(FNAMES,4,7);
   WRITELN(' REWRITE TO  ',GNAMES);
   WRITELN;
   WRITE(' ENTER PIXEL BEFORE BLACK REGION (VBL1) : ');
READLN(VBL1);
   WRITE(' ENTER PIXEL AFTER  BLACK REGION (VBL2) : ');
READLN(VBL2);
   WRITELN;
   ASSIGN(F,FNAMES);
   RESET(F);
   ASSIGN(G,GNAMES);
   REWRITE(G);
   WHILE NOT EOF(F) DO BEGIN
      FOR I:=1 TO 4 DO BEGIN
         J:=1;
         DONE:=FALSE;
         WHILE NOT DONE DO BEGIN
            READLN(F,PIX[J]);
            IF PIX[J] > -1 THEN J:=J+1
               ELSE DONE:=TRUE;
         END;
         J:=1;
         DONE:=FALSE;
```

```
            WHILE NOT DONE DO BEGIN
               WRITELN(G,PIX[J]);
               IF PIX[J] > -1 THEN
                  IF J=VBL1 THEN J:=VBL2
                  ELSE J:=J+1
               ELSE DONE:=TRUE;
            END;
         END;
         FOR I:=5 TO 8 DO BEGIN
            J:=1;
            DONE:=FALSE;
            WHILE NOT DONE DO BEGIN
               READLN(F,PIX[J]);
               WRITELN(G,PIX[J]);
               IF PIX[J] > -1 THEN J:=J+1
                  ELSE DONE:=TRUE;
            END;
         END;
      END;
      CLOSE(F); CLOSE(G);
END.
```